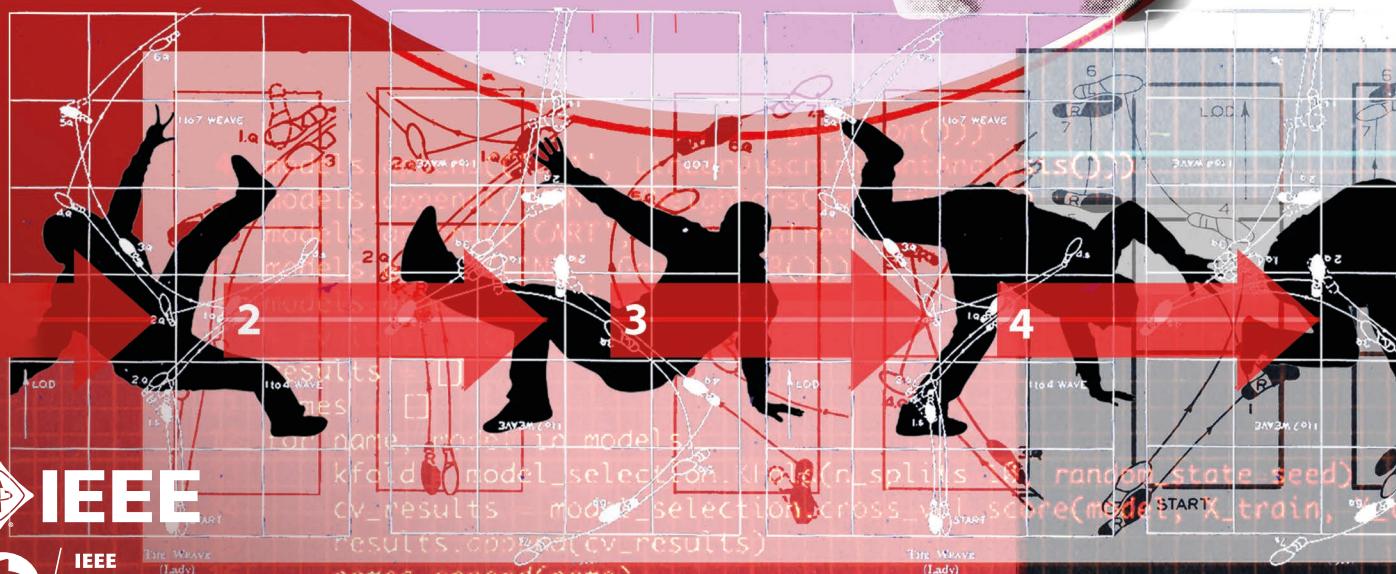


07.25

Computer

Applying AI to Software Development



IEEE Computer Society

Grants for EMERGING TECHNOLOGY ACTIVITIES

MAKE AN IMPACT | CREATE SOLUTIONS

Are you connecting the computing community with emerging technologies? Help advance emerging tech to create solutions for the betterment of humanity.

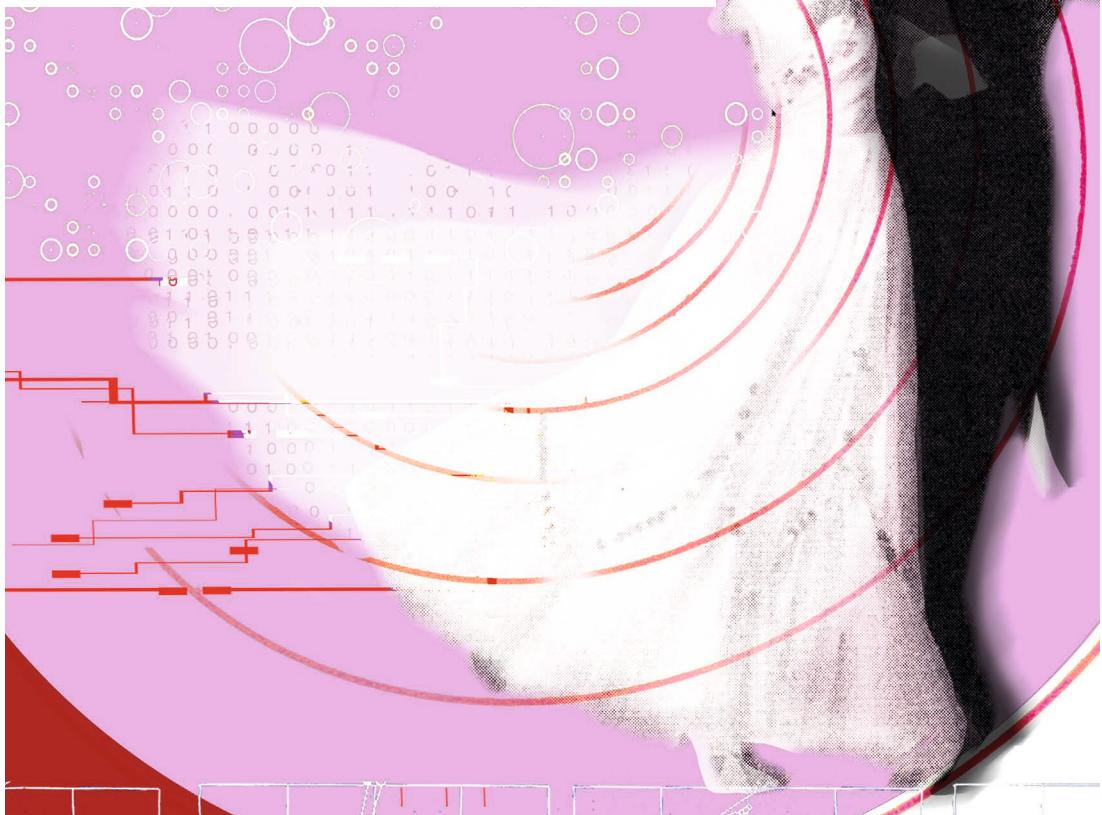
Every year, we give up to **\$50,000** in funding per project for these efforts.

Learn more at

computer.org/communities/emerging-technology-fund



Computer



17

GUEST EDITORS' INTRODUCTION

Generative AI in the Software Development Lifecycle

TRACY (TRAC) BANNON AND PHIL LAPLANTE

JULY 2025

FEATURES

20

Knowledge Monopoly Risks in
Generative AI-Assisted Software
Development Lifecycle

YANYI WU, XINYU LU, AND CHENGHUA LIN

31

Generative AI in
Software Development:
Challenges, Opportunities,
and New Paradigms for
Quality Assurance

F. MURATOVIC,
D. KEARNS-MANOLATOS,
AND A. ALIBAGE

40

AI for Productivity:
Transforming
Enterprise Software
Development

AMIT JAIN AND ROHIT TIWARI



ABOUT THIS ISSUE APPLYING AI TO SOFTWARE DEVELOPMENT

Articles explore how generative AI is reshaping software engineering.

FEATURES CONTINUED

53 Transforming Software Engineering Processes Through Generative AI: A Framework for Integration and Implementation
AYBÜKE YALÇINER, EBRU GÖKALP, AND AHMET DIKİCİ

66 ChatGPT Choreography: Discovering Developer Dialogues and Potential Software Development Lifecycle Applications
AMAN SWARAJ, SANDEEP KUMAR, AND LALIT MOHAN SHARMA

79 Leveraging LLMs for Trustworthy Software Engineering: Insights and Challenges
MARCO VIEIRA

Department

7 Elsewhere in the CS

Membership News

95 IEEE Computer Society Information

COLUMNS

4 SPOTLIGHT ON TRANSACTIONS

Pixels of Deception: How Evolutionary Algorithms Break AI Reliability
ANTONIO MASTROPAOLO

10 50 & 25 YEARS AGO

ERICH NEUHOLD

11 COMPUTING THROUGH TIME

Software Development
ERGUN AKLEMAN

14 EIC'S MESSAGE

Is Critical Thinking Critical?
JASON RUPE AND JEFFREY VOAS

91 INDUSTRY INSIGHTS

The Return of the Magic Triangle
CHRISTOF EBERT

96 EDUCATION

Computer Education in the Machine Learning Era: Intelligent Systems
BRIAN M. SADLER

100 COMPUTING'S ECONOMICS

Blockchain Standardization in Practice: Contrasting European Union and U.S. Approaches
NIR KSHETRI

110 PREDICTIONS

Predicting the Future of Supercomputing

SCOTT ATCHLEY, ROSA M. BADIA, BRONIS R. DE SUPINSKI, JOSHUA FRYMAN, DIETER KRANZLMÜLLER, SRILATHA MANNE, PEKKA MANNINEN, SATOSHI MATSUOKA, DEJAN MILOJICIC, GALEN SHIPMAN, ERIC VAN HENSEBERGEN, AND ROBERT W. WISNIEWSKI

153 OUT OF BAND

122 ARTIFICIAL INTELLIGENCE/ MACHINE LEARNING

Can AI Fix Buggy Code? Exploring the Use of Large Language Models in Automated Program Repair
LAN ZHANG, ANOOP SINGHAL, QINGTIAN ZOU, XIAOYAN SUN, AND PENG LIU

130 HUMANITY AND COMPUTING

Embracing Neurodiversity in Cybersecurity: What Lies Ahead?
BILAL NAQVI AND JOAKIM KÄVRESTAD

134 GAMES

Hype Me to the Moon! Let Me Dream Among the AI Stars! Let's Give up on That Guy Who Wants to Go to Mars ...
MICHAEL ZYDA

138 ALGORITHMS

GTS Attack: Finding Adversarial Examples With Greedy Tree Search
CHRISTOPHER CLARK AND ARMON BARTON

147 SOFTWARE ENGINEERING

Software Testing in the Generative AI Era: A Practitioner's Playbook
DOUGLAS C. SCHMIDT

153 OUT OF BAND

Generative AI and Health Care: Brief Survey
ROY RADA

157 CYBERTRUST

Bridging the Gap Between Engineering and Cybersecurity to Better Protect Critical Infrastructures
JOSEPH WEISS AND JAMES BRETT MICHAEL

161 CYBER-PHYSICAL SYSTEMS

Cyberphysical Systems in Control: Risks of Digital Transformation
DIMITRIOS SERPANOS AND PANAYOTIS ANTONIADIS

Circulation: Computer (ISSN 0018-9162) is published monthly by the IEEE Computer Society. IEEE Headquarters, Three Park Avenue, 17th Floor, New York, NY 10016-5997; IEEE Computer Society Publications Office, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720; voice +1 714 821 8380; fax +1 714 821 4010; IEEE Computer Society Headquarters, 2001 L Street NW, Suite 700, Washington, DC 20036. IEEE Computer Society membership includes a subscription to Computer magazine.

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

EDITOR IN CHIEF

Jeffrey Voas
 NIST, USA
 j.voas@ieee.org

**ASSOCIATE EDITOR IN CHIEF,
 INTERNET OF THINGS**
Joanna F. DeFranco
 The Pennsylvania State University, USA
 jfd104@psu.edu

**ASSOCIATE EDITOR IN CHIEF,
 COMPUTING PRACTICES**
Vladimir Getov
 University of Westminster, U.K.
 v.s.getov@westminster.ac.uk

**ASSOCIATE EDITOR IN CHIEF,
 COMPUTING'S ECONOMICS**
Nir Kshetri
 The University of North Carolina at
 Greensboro, USA
 nbkshetr@uncg.edu

**ASSOCIATE EDITOR IN CHIEF,
 CYBERTRUST**
James Bret Michael
 Naval Postgraduate School, USA
 bmichael@nps.edu

**2025 IEEE COMPUTER SOCIETY
 PRESIDENT**
Hironori Washizaki
 Waseda University, Tokyo, Japan
 washizaki@waseda.jp

AREA EDITORS

BIG DATA
Domenico Talia
 University of Calabria, Italy

CLOUD COMPUTING
Schahram Dustdar
 TU Wien, Austria

COMPUTING FUNDAMENTALS
Abderazek Ben Abdallah
 The University of Aizu, Japan

Joanna F. DeFranco
 The Pennsylvania State University, USA

Jia Zhang
 Southern Methodist University, USA

CYBERSECURITY
Rick Kuhn
 NIST, USA

M S Raunak
 NIST, USA

EMBEDDED COMPUTING
Marilyn Wolf
 University of Nebraska, USA

EMERGING TECHNOLOGIES
Irena Bojanova
 NIST, USA

Elena Loukoianova
 International Monetary Fund, USA

HIGH-PERFORMANCE COMPUTING
Vladimir Getov
 University of Westminster, U.K.

INTERNET OF THINGS
Michael Beigl
 Karlsruhe Institute of Technology, Germany

SECURITY AND PRIVACY
James Bret Michael
 Naval Postgraduate School, USA

SOCIAL-PHYSICAL-CYBER SYSTEMS
Mike Hinchev
 University of Limerick, Ireland

SOFTWARE ENGINEERING
Benoit Baudry
 KTH Royal Institute of Technology, Sweden

Christof Ebert
 Vector Consulting Services/University of Stuttgart, Germany

Phil Laplante
 The Pennsylvania State University, USA

COLUMN AND DEPARTMENT EDITORS

AFTERSHOCK
Solom Heddaya
 Heddaya Projects LLC, USA

ALGORITHMS
Doron Drusinsky
 Naval Postgraduate School, USA

ARTIFICIAL INTELLIGENCE/MACHINE LEARNING
Hsiao-Ying Lin
 Huawei France, France

COMPUTING ARCHITECTURES
Timothy Jones
 University of Cambridge, U.K.

Robert Mullins
 University of Cambridge, U.K.

COMPUTING'S ECONOMICS
Nir Kshetri
 The University of North Carolina at Greensboro, USA

COMPUTING THROUGH TIME
Ergun Akleman
 Texas A&M Univ., USA

CYBER-PHYSICAL SYSTEMS
Dimitrios Serpanos
 University of Patras, Greece

CYBERTRUST
James Bret Michael
 Naval Postgraduate School, USA

DATA
Norita Ahmad
 American University of Sharjah, United Arab Emirates

EDUCATION
Preeti Chauhan
 Google, USA

GAMES
George Hurlburt
 U.S. Federal Service (Retired), USA

MEMORY AND STORAGE
Sorel Reisman
 California State University, USA

MEMPHIS
Conrad James
 Sandia National Laboratories, USA

HUMANITY AND COMPUTING
Domenico Talia
 University of Calabria, Italy

INDUSTRY INSIGHTS
Christof Ebert
 Vector Consulting Services, Germany

INTERNET OF THINGS
Joanna F. DeFranco
 The Pennsylvania State University, USA

IT INNOVATION
Mark Campbell
 EVOTEK, USA

MICROELECTRONICS
Tom Coughlin
 Coughlin Associates, USA

OPEN SOURCE
Dirk Riehle
 Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

OUT OF BAND
Hal Berghel
 University of Nevada, Las Vegas, USA

PREDICTIONS
Dejan Milojicic
 Hewlett Packard Labs, USA

SOFTWARE ENGINEERING
Phil Laplante
 The Pennsylvania State University, USA

SPOTLIGHT ON TRANSACTIONS
Antonio Mastropao
 College of William and Mary, USA

STANDARDS
Jyotika Athavale
 Synopsys, Inc., USA

50 & 25 YEARS AGO
Erich Neuhold
 University of Vienna, Austria

ADVISORY PANEL

Carl K. Chang (EIC Emeritus), Iowa State University, USA
 Sumi Helal (EIC Emeritus), University of Bologna, Italy
 Keith Miller, retired, USA
 Bill Schilit, Google, USA
 George K. Thiruvathukal, Loyola University Chicago, USA
 Ron Vetter (EIC Emeritus), University of North Carolina Wilmington, USA
 Alf Weaver, University of Virginia, USA

CS PUBLICATIONS BOARD

Charles (Chuck) Hansen (VP for Publications), Irena Bojanova, Greg Byrd, Sven Dickinson, David Ebert, Minyi Guo, Lizy K. John, Joaquim Jorge, Daniel S. Katz, Klaus Mueller, San Murugesan, Jaideep Vaidya; Ex officio: Hironori Washizaki, Melissa Russell, Robin Baldwin

COMPUTER STAFF

Journals Production Manager

Joanna Gojlik
 j.gojlik@ieee.org

Cover Design

Nanette Hoogslag

Peer Review Administrator

computer-ma@computer.org

Periodicals Portfolio Specialist

Priscilla An

Periodicals Operations Project Specialist

Christine Shaughnessy

Compliance Manager

Jennifer Carruth

Periodicals Portfolio Senior Manager

Carrie Clark

Senior Advertising Coordinator

Debbie Sims

Director of Periodicals & Special Projects

Robin Baldwin

IEEE Computer Society

Membership Director

Erik Berkowitz

IEEE Computer Society Executive Director

Melissa Russell

IEEE PUBLISHING OPERATIONS

Senior Director, Publishing Operations

Dawn M. Melley

Director, Editorial Services

Kevin Lisankie

Director, Production Services

Peter M. Tuohy

Associate Director, Digital Assets & Editorial Support

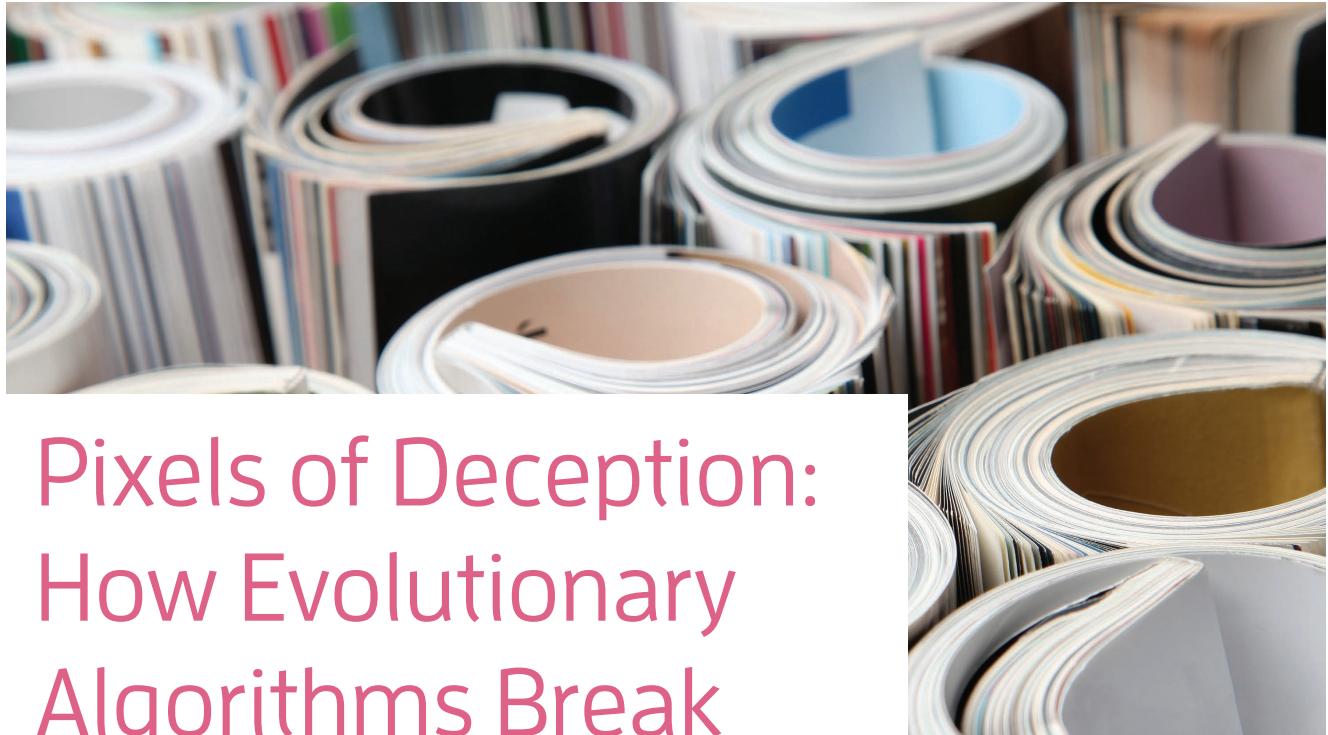
Neelam Khinvasara

Senior Manager, Journals Production

Patrick Kempf

Digital Object Identifier 10.1109/MC.2025.3561964

Permission to reprint/republish this material for commercial, advertising, or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by writing to the IEEE Intellectual Property Rights Office, 445 Hoes Lane, Piscataway, NJ 08854-4141 or pubs-permissions@ieee.org. Copyright © 2025 IEEE. All rights reserved. IEEE prohibits discrimination, harassment, and bullying. For more information, visit www.ieee.org/web/aboutus/whatis/policies/p9-26.html.



Pixels of Deception: How Evolutionary Algorithms Break AI Reliability

Antonio Mastropaoletti, William & Mary

The article reviews the evolutionary universal perturbation attack, a technique that uses genetic algorithms to craft tiny, invisible noise patterns that can mislead deep neural networks without needing internal model access.

Deep neural networks (DNNs) are powerful machine learning models inspired by the human brain. They often match or even surpass human performance in tasks like recogniz-

ing images and identifying objects. Despite these impressive abilities, DNNs are affected by a surprising weakness: they can be easily fooled by tiny, almost invisible changes to their input data. These minor tweaks, known as adversarial perturbations, can trick a model into making completely incorrect decisions.

Digital Object Identifier 10.1109/MC.2025.3566179
Date of current version: 27 June 2025

Although adversarial perturbations may initially appear to be a theoretical concern, Goodfellow et al.² provided a compelling real-world demonstration of their effectiveness. In their experiment, they added a carefully crafted, visually imperceptible noise pattern to an image of a panda. While the altered image appeared unchanged to human observers, the neural network was deceived into classifying it as a gibbon with 99.7% certainty.

Even more concerning is the discovery that attackers don't need to design unique perturbations for every image. Universal adversarial perturbations have emerged as fixed noise patterns capable of deceiving artificial intelligence (AI) models across a wide range of images. In practice, an attacker can craft a single, tiny, and invisible "mask" that consistently disrupts classification across almost any image or video frame. These



universal perturbations expose fundamental vulnerabilities in deep learning models, highlighting just how fragile they can be.

Given the current state of affairs, we must confront a critical question: how can we trust AI when its decision-making can be systematically manipulated?

This month, we review a pioneering paper in the field of adversarial attack generation. Specifically, Gupta et al.³ introduce the evolutionary universal perturbation attack (EUPA), a novel method that applies principles of Darwinian evolution to craft highly effective adversarial attacks. EUPA simulates natural selection to iteratively evolve tiny perturbations—changes humans cannot detect—that consistently expose AI blind spots. At the heart of EUPA is a genetic algorithm that maintains a population of potential noise patterns. Each pattern's effectiveness (or fitness^a) is judged by how many images it successfully causes the model to misclassify, while remaining visually imperceptible. The best-performing patterns are combined and slightly altered to form the next generation, gradually creating highly effective yet subtle adversarial perturbations. Notably, this approach operates without any prior knowledge of the model's internal structure, making it an especially elusive form of input–output attack.^b

When researchers tested EUPA on a widely used image recognition model (GoogleNet) with the popular ImageNet data set,¹ the results were striking. The evolved perturbation significantly

reduced the model's accuracy—from nearly perfect detection to roughly 44%. Remarkably, the affected images appeared completely unchanged to human viewers. Simply put, EUPA identified a nearly invisible “mask” capable of disrupting a sophisticated AI system across diverse images.

for humans but highly sensitive for machine learning models.

Reflecting on what we have explored this month, it is clear that AI models powered by DNNs can be fooled far more easily than most of us might expect—often in ways humans would never notice. As AI becomes more deeply woven into our

Universal adversarial perturbations have emerged as fixed noise patterns capable of deceiving artificial intelligence models across a wide range of images.

We invite readers to take a moment to examine [Figure 1](#) and observe how EUPA can subtly alter the original picture (leftmost) without introducing significant perceptual flaws (rightmost). This subtle manipulation reveals the fundamental differences between how humans and neural networks approach classification tasks such as image recognition, which is resilient to minor changes

daily lives—from automating recurrent tasks to securing our identities—these hidden vulnerabilities highlight why building trustworthy and resilient systems is more important than ever.

Techniques like EUPA mark a critical shift in our digital landscape. Unlike traditional security threats, where attackers and defenders often operate under similar conditions, universal adversarial approaches



Figure 1. Example inspired from Gupta et al.,³ illustrating how EUPA injects universal noise into images.

^aIn evolutionary algorithms, fitness measures how well a solution achieves its goal—in this case, how successfully it tricks the AI model.

^bAn input–output attack refers to a scenario where the attacker has no direct access to the AI model's internal workings or parameters. In particular, they only interact with the model by providing inputs and observing the outputs, making it significantly harder to detect and defend against.

allow attackers to create widespread disruption with tiny, almost invisible changes. What is at stake here is the very foundation of trust we place in AI systems—trust that they will perceive,

to threats and instead focuses on anticipating and addressing the hidden weaknesses within AI models. History shows that real progress, whether in web technology or automotive safety,

database," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Piscataway, NJ, USA: IEEE Press, 2009, pp. 248–255, doi: [10.1109/CVPR.2009.5206848](https://doi.org/10.1109/CVPR.2009.5206848).

2. I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, arXiv:1412.6572.
3. N. Gupta, M. Khosravy, A. Pasquali, and O. Witkowski, "Deep learning security breach by evolutionary universal perturbation attack (EUPA)," IEEE Trans. Artif. Intell., vol. 5, no. 11, pp. 5655–5665, Nov. 2024, doi: [10.1109/TAI.2024.3429473](https://doi.org/10.1109/TAI.2024.3429473).

EUPA simulates natural selection to iteratively evolve tiny perturbations—changes humans cannot detect—that consistently expose AI blind spots.

decide, and act reliably even under unpredictable conditions. If such systems can be derailed by imperceptible tweaks, the risks extend far beyond technical failure, potentially affecting public safety, privacy, and confidence in the technologies that increasingly shape our lives.

has always come from confronting vulnerabilities head-on. Similarly, by studying methods like EUPA, we can better understand how AI systems fail, and in doing so, make them better, more reliable, and ultimately more deserving of the trust we increasingly place in them. □

REFERENCES

1. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image

ANTONIO MASTROPAOLO is an assistant professor at William & Mary, Williamsburg, VA 23185 USA. Contact him at amastropao@wm.edu.

This new reality demands a different mindset—one that moves beyond simply reacting

The advertisement features a man with dark hair and a beard, wearing glasses and a blue sweater, looking upwards. Numerous lightbulbs are drawn around his head, symbolizing ideas and innovation. The background is a textured gray. In the upper left, a yellow box contains the text "IEEE COMPUTER SOCIETY Call for Papers". In the upper right, another yellow box contains the text "Build your authority in the industry with exposure to a global network of 350K+ computing professionals". In the bottom right, there is a QR code with the text "GET PUBLISHED" and the website "www.computer.org/cfp". The bottom right corner also features the IEEE Computer Society logo and the IEEE logo.

Digital Object Identifier 10.1109/MC.2025.3573113

ELSEWHERE IN THE CS



Computer Highlights Society Magazines

The IEEE Computer Society's lineup of 11 peer-reviewed technical magazines covers cutting-edge topics ranging from software design and computer graphics to Internet computing and security, from scientific applications and machine intelligence to visualization and microchip design. Here are highlights from recent issues.



Bridging the Gender Gap in Computing: Insights From Latin America With a Focus on Brazil

The gender gap in digital skills and computing persists globally despite initiatives to include women and girls. This October–December 2024 *Computing in Science & Engineering* article discusses recent initiatives in Latin America, with a special focus on Brazil, that aim to increase women's access, skills, and leadership in technology.



Mechanical Numeracy: Thinking With the Soroban Abacus in Modern Japan

The author of this article, featured in the January–March 2025 issue of *IEEE Annals of the History of Computing*, discusses the soroban, a traditional arithmetic tool whose life in modern Japan reflected an inherently material dimension of computational numeracy. Despite the introduction of Western-style pen-and-paper arithmetic, it persisted as the standard tool of computing in households, businesses, and bureaucracy, driving the Taylorist training of human calculators who could apply its commands with corporal

discipline and the design of mechanical calculators that simplified its procedures.



Augmented Reality Art Museum Mobile Guide for Enhancing User Experience

The advancement of augmented reality (AR) technology provides means for next-generation art museum tour guides. In this study, featured in the January/February 2025 issue of *IEEE Computer Graphics and Applications*, the authors develop an AR navigation and multimedia guide for mobile devices and head-mounted displays. Two case studies are conducted to evaluate the proposed system. This research contributes to the field of AR technology and cultural heritage education by offering a mobile application for indoor navigation and artwork exploring, allowing art museum visitors to actively engage with the exhibits and enhance their understanding while maintaining their interests.



Boundaries Matters: A Novel Multibranch Semisupervised Semantic Segmentation Method

In recent years, semisupervised semantic segmentation (SSS) research has been progressing rapidly. Existing methods usually ignore the classification of detailed pixels, such as boundaries, resulting in degraded segmentation performance. To overcome this challenge, the authors of this article featured in the March/April 2025 issue of *IEEE Intelligent Systems* propose a new multibranch SSS framework, BoundaryMatch, which combines image- and feature-level perturbation branches with a boundary detail guidance branch, all utilizing a shared encoder.



UniTwin: Pushing Universal Digital Twins Into the Clouds Through Reconfigurable Container Environments

The increasing intersection of physical and digital spaces through intelligent digital twins (DTs) is revolutionizing real-world facilities and processes. This article from the January/February 2025 issue of *IEEE Internet Computing* presents UniTwin, a comprehensive framework developed for the deployment and aggregation of cloud-native universal DTs in containerized environments. The authors contribute the design, implementation, and validation of the UniTwin framework and demonstrate its effectiveness and potential for future diversified applications.



UCle Standard: Enhancing Die-to-Die Connectivity in Modern Packaging

Today, packaging plays a crucial role in determining chip performance in systems. As transistor sizes shrink and chip sizes grow to meet increasing processing demands, wafer yield drops. Connecting smaller chips within a package or using multidie designs has gained attention as it improves yield, enables intellectual property reuse, enhances performance, and lowers costs. This article featured in the January/February 2025 issue of *IEEE Micro* focuses on achieving effective die-to-die connectivity using organic and advanced packaging. These patterns have been validated through simulations and the fabrication of the test chip.



Development of an Image Encryption Algorithm Based on Compressed Sensing and Chaotic Mapping

This October–December 2024 *IEEE MultiMedia* article introduces a new 2-D hyperchaotic map design based on 2-D discrete mapping, featuring two positive Lyapunov exponents, enhanced randomness, and initial offset boosting. On this basis, an image encryption algorithm is developed, grounded in chaotic systems, compressed sensing theory, and scrambling-diffusion operations. Validation through a histogram, correlation coefficient, and information entropy operations proves

the algorithm's effective image encryption. Robustness is verified through plaintext and differential attacks, and metrics such as peak signal-to-noise ratio and structural similarity index affirm the algorithm's strong restorative capability.



An App-Assisted Frontend of Robot Gait Training System for Lower Limb Rehabilitation

Robot-assisted gait training (RAGT) systems merge traditional rehabilitation benefits with precise therapy control. This article, featured in the October–December 2024 issue of *IEEE Pervasive Computing*, introduces MRGtalk, an app-assisted frontend of RAGT system for lower limb rehabilitation due to strokes and neurological disorders, targeting elderly individuals for enhanced limb activity and physical fitness. MRGtalk employs pervasive computing for lower limb rehabilitation, emphasizing muscle strength improvement and walking ability enhancement.



Contextualizing Formal Verification for Systems Security Engineering

While formal methods are recognized as means to establish the trustworthiness of systems, their practical adoption for verifying system-level security is limited. By reflecting on system development approaches, the authors of this article, featured in the March/April 2025 issue of *IEEE Security & Privacy*, identify and exemplify pertinent challenges and suggest directions to facilitate the effective, scalable use of formal methods when designing systems.



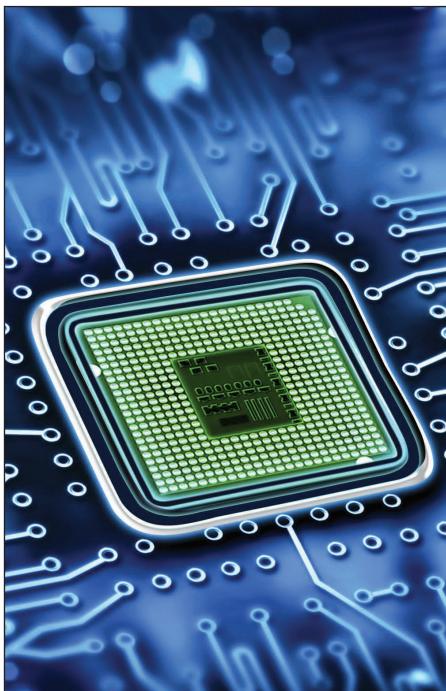
Leveraging Creativity as a Problem-Solving Tool in Software Engineering

In this article, the author of this article from the May/June 2025 issue of *IEEE Software*, ports results from creativity research in the field of cognitive psychology to the field of software engineering. After all, programming is a highly creative endeavor. The author explores how to leverage creativity as a practical problem-solving tool to wield for software developers.

Generative Artificial Intelligence in Financial Services: Opportunities, Challenges, and Cyberthreats

Generative artificial intelligence (GAI) is transforming the financial services sector, offering innovative solutions and equipping the industry with cutting-edge tools to drive value

creation and operational efficiency. In this article, featured in the March/April 2025 issue of *IT Professional*, the authors assess the economic benefits of GAI technology, identify institutional challenges associated with its adoption, examine key regulatory initiatives, and outline a risk-mitigation framework. Their proposed solution to mitigate institutional challenges is twofold: 1) actionable institutional risk management safeguards, and 2) a baseline safety net for GAI implementation in the financial services sector. □



IEEE TRANSACTIONS ON COMPUTERS

Call for Papers

Publish your work in the IEEE Computer Society's flagship journal, *IEEE Transactions on Computers*. The journal seeks papers on everything from computer architecture and software systems to machine learning and quantum computing.



Learn about calls for papers
and submission details at
www.computer.org/tc



Digital Object Identifier 10.1109/MC.2025.3575260

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

Reuse Rights and Reprint Permissions: Educational or personal use of this material is permitted without fee, provided such use: 1) is not made for profit, 2) includes this notice and a full citation to the original work on the first page of the copy, and 3) does not imply IEEE endorsement of any third-party products or services. Authors and their companies are permitted to post the accepted version of IEEE-copyrighted material on their own web servers without permission, provided that the IEEE copyright notice and a full citation to the original work appear on the first screen of the posted copy. An accepted manuscript is a version which has been revised by

the author to incorporate review suggestions, but not the published version with copyediting, proofreading, and formatting added by IEEE. For more information, please go to: http://www.ieee.org/publications_standards/publications/rights/paperversionpolicy.html. Permission to reprint/republish this material for commercial, advertising, or promotional purposes or for creating new collective works for resale or redistribution must be obtained from IEEE by writing to the IEEE Intellectual Property Rights Office, 445 Hoes Lane, Piscataway, NJ 08854-4141 or pubs-permissions@ieee.org. Copyright © 2025 IEEE. All rights reserved.

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

50 & 25 YEARS AGO



EDITOR ERICH NEUHOLD 

University of Vienna
erich.neuhold@univie.ac.at



JULY 1975

<https://www.computer.org/cslm/magazine/co/1975/07>

Guest Editor: Computers in Cardiology; Kenneth M. Kempner (p. 28): “The computer has, over the past decade, been applied to virtually all phases of cardiovascular patient care, diagnostics, and research. ... an International Conference on Computers in Cardiology was convened in Bethesda, Maryland on October 2-4, 1974. The conference, which attracted engineers, computer scientists, and medical professionals from 11 countries.” (p. 29) “Five papers presented at the Conference are being reprinted in this issue of Computer as typical examples of current trends in the field.” [Editor’s note: It is, at least for me, interesting to see the wide use of computer technology in medicine in 1975. Cardiology, described in this issue, is just an example. However, besides the computer system-related descriptions, all the papers presume quite some medical knowledge of the reader.]

Continuous Monitoring of Physiologic Variables with a Dedicated Minicomputer; L. J. Thomas, Jr. et al. (p. 30) “The system provides for continuous processing of ECG; arterial, central venous, left atrial, and airway pressures; temperature; respiratory gas flow; and urine flow. Derived values include heart rate; systolic, mean, and diastolic arterial pressures; mean central venous and left atrial pressures; peak inspiratory airway pressure; respiratory rate; tidal volume; total thoracic compliance; rectal temperature; and urinary output.” [Editor’s note: An impressive array of data being collected and analyzed. The article provides a detailed description of the system, its advantages but also flaws that were encountered during 18 months of use in a four-bed intensive care unit.]

Problems in Acquisition and Representation of Coronary Arterial Trees; C. Frank Starmer et al. (p. 36) “The nature of the contractions of the heart wall has long been recognized as indicative of the presence and degree of atherosclerotic

lesions in the coronary arteries. ... This paper is a report of preliminary efforts to process trees by computer and a discussion of some of the problems involved in this effort.” (p. 41) “Thus, the systems described above are just a beginning in the investigation ... the significance of the geometry of the coronary tree to clinical medicine.” [Editor’s note: The system and the processing of data is described in detail but, as the previous conclusion says, there has not been any analysis that would allow the use these data for diagnostic purposes.]

On-Line Analysis of Exercise Electrocardiograms Maarten L. Simoons et al. (p. 43) “Also, an on-line system can provide information on ECG abnormalities immediately when these occur. This will enhance the safety of exercise testing and may shorten the duration of the diagnostic procedure. ... At our department a system for processing of orthogonal, three-lead XECGs using a PDP-8E computer has been developed.” [Editor’s note: A detailed and interesting discussion of the system and its use is included and shows the benefits of the continued analysis of Exercise Electrocardiograms.]

Small Computer Processing of Ambulatory Electrocardiograms; John W. Fitzgerald et al. (p. 48) “The system, which is described below, was designed within the following constraints: 1) construction with easily obtainable components, 2) capable of qualitatively describing all arrhythmias occurring during the period of recording, and 3) capable of quantifying ventricular arrhythmias with an acceptable error rate.” (p. 52) “There have been 410 12-hour, and 436 24-hour tapes processed with the system, including 298 tapes of clinical patients evaluated for arrhythmias, syncope, and drug response.” [Editor’s note: A very detailed description of the hardware and software used, mostly dealing with technological constraints encountered 50 years ago. The detailed evaluation unfortunately does not include an evaluation of the benefits the system has offered.]

Automated Video-Angiocardiographic Image Analysis; Paul H. Heintzen et al. (p. 55) “We will concentrate in particular on some aspects of automated image processing,

i.e., the analysis of the size, shape, and contraction pattern of the ventricles from video-angiograms." (p. 59) "If the great variety of heart positions and long axis orientation are not taken into consideration in volume determinations, the

accuracy is generally reduced." [Editor's note: A very detailed description of all the variations one has to observe for the proper evaluation, for example, of the volume of the Left Ventricular, together with a description of the equipment used.]

COMPUTING THROUGH TIME SOFTWARE DEVELOPMENT



BY ERGUN AKLEMAN 
ERGUN.AKLEMAN@GMAIL.COM

SOFTWARE DEVELOPMENT
BEFORE GENERATIVE AI:
AT A TECH COMPANY
IN THE TWENTIETH CENTURY



SOFTWARE DEVELOPMENT
AFTER GENERATIVE AI:
AT A TECH COMPANY
IN THE TWENTY FIRST CENTURY



In the spirit of this month's computer theme on generative AI in the software development lifecycle (SDLC), I created the two panels of this cartoon primarily using generative AI – specifically, ChatGPT. Of course, the illustrations weren't immediately usable as generated. The left panel required only minor adjustments: I modified the facial features slightly to better match my style and added a few sweat droplets to enhance the expression. The second panel posed more challenges.

For example, I specifically asked for legs resting on the table, but the result had incorrectly drawn crossed legs and feet oddly floating in the air. Several other drawing inconsistencies also needed correction. Nevertheless, all these issues were easy to fix. Overall, using generative AI is far more efficient than drawing both panels entirely from scratch. It offers a solid starting point and allows cartoonists to focus on refining rather than reinventing.

50 & 25 YEARS AGO

Special Tutorial; BAYESIAN STATISTICS: MATHEMATICAL TOOLS FOR COMPUTERS; D. E. Raeside (p. 65): "This tutorial article deals with a branch of statistics known as "Bayesian statistics." To better understand the underlying philosophy of Bayesian statistics, let us contrast its outlook with that of "classical statistics." [Editor's note: A very readable introduction to Bayesian Statistics.]

JULY 2000

<https://www.computer.org/csdl/magazine/co/2000/07>

Video Streams into the Mainstream; George Lawton (p.12): "Graphics, animation, and audio technologies have already proliferated on the Internet. However, streaming video—in which viewers can begin watching content almost as soon as it starts downloading—is only now developing as a mainstream technology. ... A typical 56-Kbps modem connection can provide only the traditional small video window." (p. 15) "However, streaming-video content itself will have to be compelling for user interest in the technology to grow." [Editor's note: Interestingly, 25 years ago streaming videos was not quite there. Even when looking at the future the article misses the tremendous growth in network speed and the interest found in "low quality" videos as on YouTube and others.]

Broadband Fixed Wireless Travels the Last Mile; Sixto Ortiz Jr. (p. 18): "To meet this challenge, service providers are beginning to employ broadband fixed wireless technologies, such as LMDS (local multipoint distribution system) and MMDS (multichannel multipoint distribution system)." (p. 19) "However, LMDS's short-wavelength signals cannot pass around or penetrate objects like buildings, walls, or thick foliage." [Editor's note: Despite the enthusiasm but also caution expressed in the article, these technologies did not reach the universal usage predicted. Of course, with 5G the problems mentioned about the penetration and the utilization of the high speed came back in force.]

News Briefs; Ed: Anne C. Lear (p. 22ff): "Do E-Business Patents Help or Hurt? A controversy is growing over the practice by many companies of patenting aspects of their e-commerce Web sites, including methods for conducting business electronically. ... However, Godici acknowledged that examiners need a stronger prior-art database, particularly for business methods." [Editor's note: The controversy shown in this article, of course, has continued until today. Many legal cases illustrate that fact.] ... **"Small Wonder: MEMS Will Power Optical Internet:** On current networks, signals must be converted to electronic signals at switches and back into optical signals when reentering the fiber portion of the network. Purely optical routing could help carriers cut costs, reduce power consumption, and eliminate many of the electronic bottlenecks that limit network speeds." [Editor's note: Both technologies purely optical and optical-electronic MEMS became essential for fiber-optic communication, but many other applications are around.]

New Protocol Simplifies B2B Communication: The Simple Object Access Protocol (SOAP) provides a mechanism for integrating services on the Internet and/or intranets, regardless of operating system, object model, or programming language." [Editor's note: A very successful protocol just 25 years old!]

SPEC CPU2000: Measuring CPU Performance in the New Millennium; John L. Henning (p. 28): "The SPEC consortium's mission is to develop technically credible and objective benchmarks so that both computer designers and purchasers can make decisions on the basis of realistic workloads." (p. 30) "The process worked well: SPEC gained objective data about candidate benchmarks' I/O, cache and main memory behavior, floating-point operation mixes, branches, code profiles, and code coverage." [Editor's note: A rather interesting and detailed description about the building of the Spec CPU2000 benchmark. The consortium is still around and has branched out into a number of other areas where their benchmarks are supposed to "level the field."]

The Wireless Internet: Promises and Challenges; David J. Goodman (p. 36): "Despite widespread optimism, mobile wireless computing has not taken off. Four major obstacles have inhibited its widespread use: • low transmission rates • costs too much • consumes too much power • interfaces poorly match the needs of people." (p. 41) "By thus analyzing a wide range of information types, we can match information services and information technology to the nature of information and the needs and budgets of people who use information services." [Editor's note: The article is mostly interesting that it seems to miss the real development that took place after 2000. It predicts the success of Wireless Application Protocol (WAP) and Infostations. But it fails to see the success of 3G (later 4G and now 5G) and the move of data processing from laptops to tablets and now to smart phones.]

PicoRadio Supports Ad Hoc Ultra-Low Power Wireless Networking; Jan M. Rabaey et al. (p. 42): "PicoRadio supports the assembly of an ad hoc wireless network of self-contained mesoscale, low-cost, low-energy sensor and monitor nodes." (p. 43) "We envision a power-dissipation level below 100 microwatts, as this would enable selfpowered nodes using energy extracted from the environment—an approach called energy-scavenging or harvesting." (p. 48) "As we show, the only way to implement an ultralow power node is by optimizing all layers of the protocol." [Editor's note: A very interesting and detailed article that explains how a self-powered network could be built. Of course, even today many aspects of self-powered systems are subject of intensive research, but a large number of applications also exist. See, for example, <https://www.sciencedirect.com/topics/engineering/self-powered>.]

Reliability and Survivability of Wireless and Mobile Networks; Andrew P. Snow et al. (p. 49): "Failures not only affect current voice and data use but could also limit emerging wireless applications such as e-commerce and

high-bandwidth Internet access." (p. 55) "So far, users seem willing to expect less, but with increasing usage, diversity, and emphasis on subscriber services, expectations will change as users compare wireless and mobile networks with the high degree of dependability of (Ed: wired) telephone and Internet access." [Editor's note: A very interesting, detailed, and worth to read analysis of possible failures and proposals on how to remedy them. However, even today users are, or better need to be, quite tolerant in accepting mobile and wireless service problems.]

Using Paths to Measure, Explain, and Enhance Program Behavior; Thomas Ball et al. (p. 57): "Paths can reveal a program's dynamic behavior and uncover patterns of path locality that can be exploited to increase program performance. The authors explore several methods for doing so." (p. 64) "In the compiler, debugging, and testing fields, program paths are coming into use and much work remains." [Editor's note: An interesting article that explains how path analysis can be used to improve software performance.]

A Staged Model for the Software Life Cycle; Václav T. Rajlich et al. (p. 66): "We describe a new view of the software life cycle in which maintenance is actually a series of distinct stages, each with different activities, tools, and business consequences. ... SOFTWARE STAGES: • Initial development • Evolution • Servicing • Phaseout • Closedown. A variation of this process is the versioned staged model, the backbone of which is the evolution stage." [Editor's note: The article then describes the evolutionary behavior of those stages. I believe that, in practice, many companies have and are actually using this model.]

How Perspective-Based Reading Can Improve Requirements Inspections; Forrest Shull et al. (p. 73): "Perspective-based reading provides a set of procedures that can help developers solve software requirements inspection problems." (p. 75) "PBR operates under the premise that different information in the requirements is more or less important for the different uses of the document." [Editor's note: The very detailed article in a way assumes that PRB of the requirements specification solves all (most?) problems and errors that may arise in later stages of the development. You may read it, but I personally am not convinced.]

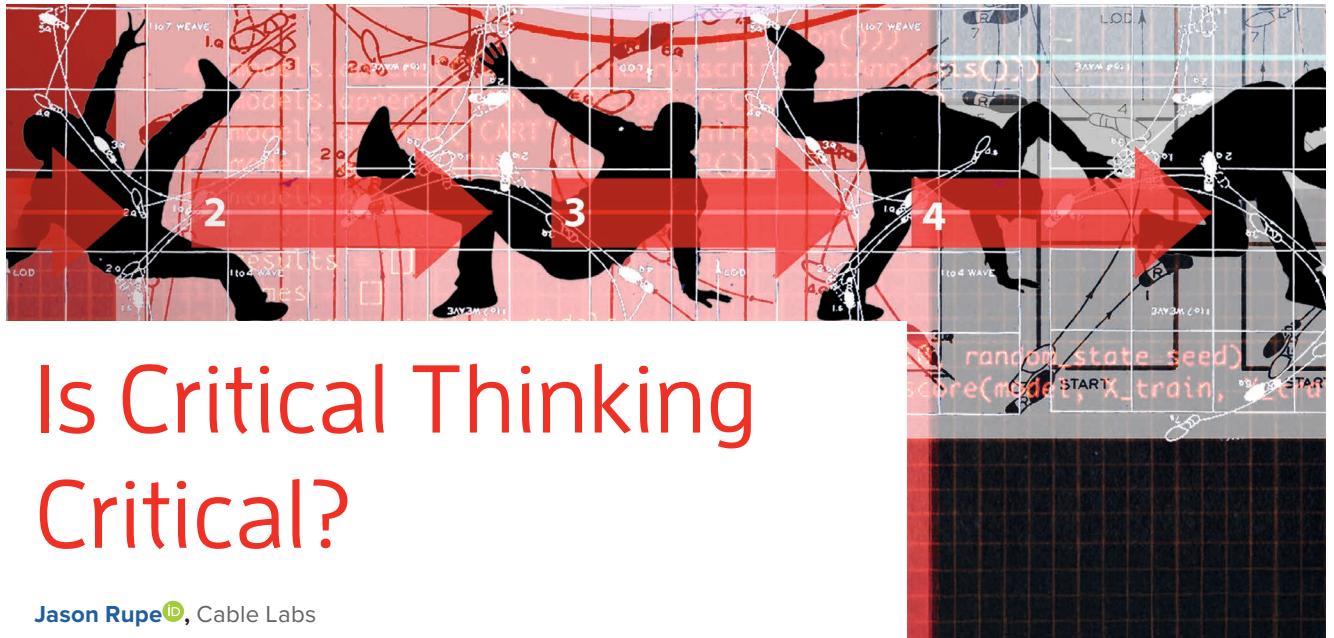
Fashioning a Foundation for the Computing Profession; Neville Holmes (p. 97): All three of these collective nouns — profession, craft, and trade— denote a college of practitioners who share a body of skills and knowledge. But each differs in the responsibilities it places on its members." [Editor's note: A very nice description of the differences between these three member societies. The Computer Society fits very well into the description of PROFESSION.]

Requirements that Handle IKIWISI, COTS, and Rapid Change; Barry Boehm (p. 99): "The recent developments of IKIWISI (I'll know it when I see it), COTS (commercial-off-the-shelf) software, and the increasingly rapid change in information technology have combined to unsettle the foundations of the old airtight-requirements approach." [Editor's note: Again, an article about different aspects of requirement specifications to handle KIWISI, COTS, and Rapid Change.]

Obelix Searches Internet Using Customer Data; Veljko Milutinovic et al. (p. 104): "The ranking should also consider the satisfaction of previous customers purchasing from those sites, customer profiles, and customer behavior. The Obelix search engine uses reconfigurable technology to apply customer satisfaction data obtained from the Internet service provider infrastructure to refine its search criteria." (p. 107) "Future research should focus on using reconfigurable technology to refine searches on the basis of customer profiles and customer behavior." [Editor's note: In interesting article, especially about the interplay between software and reconfigurable hardware. Utilization of user data, of course, is now common in search engines but reconfigurable hardware, for example, field-programmable gate arrays (FPGAs), is, to my knowledge, not widely used in computing farms.]

Managing System and Active-Content Integrity; John R. Michener et al. (p. 108): "Our method employs an executable content loader (which we call a strong loader) and a short-lived configuration management file to address the software aging problem." [Editor's note: The article discusses the many problems of loading executable new software or updates of old one and proposes a solution that relies on the loader as being an integral part of the operating system.]

The Web Fosters a Rust-Belt Renaissance; Ted Lewis (p. 112): "With the exception of General Electric, the old-line industrial companies slipped from the minds of most investors. Most have been barely able to maintain a price/earnings ratio of 10 to 1, while dot-com valuations have soared to 100 to 1 and more. The old economy is striking back with a vengeance against the brash upstarts that watched in horror as their own market cap valuations plunged in the crash of 2000." (p. 111) "Fortune 1000 companies, which mostly include big, old-economy companies, account for more than one third of all intercompany commerce in the US." [Editor's note: Of course, e-commerce continued to grow rapidly over the years. But looking at today's statistics there is a wide spread in the numbers. I believe that is due to the facts that the term "e-commerce" is actually poorly defined. However, the stock prices still have this odd behavior mentioned in this 25-year-old article.]



Is Critical Thinking Critical?

Jason Rupe^{ID}, Cable Labs

Jeffrey Voas^{ID}, IEEE Fellow

Critical thinking—the process of using evidence, observations, and logical arguments to evaluate information, make decisions, or draw conclusions—has long been an important staple of science and engineering. This is particularly true in computing, a field in which emerging technologies increasingly require this skill—both for the producer and the user.

Critical thinking (CT) may be a generally unfamiliar term, but it is growing in importance, so we thought now might be a good time to revisit it. Maybe you learned about it indirectly as part of your studies in computer science and engineering. As society embarks on several new technological fronts, CT is becoming more critical. This is particularly relevant in the computer industry, which is both impacted by and impacting our ability to be critical thinkers.

Digital Object Identifier 10.1109/MC.2025.3554307
Date of current version: 27 June 2025

A CT SCAN

CT is the process of using evidence, facts, observations, and logical arguments to make decisions, choices, or conclusions. At the heart of all of this is trust—and its partner verification.

The value of evidence, the trustworthiness of facts, and the believability of observations are all linked to the quality of the information that is the foundation of CT. Quality is assured through scrutiny, inspection, verification, and confirmation.

With new technologies, we are exposed to much more information with less time to research its trust-

worthiness. We have greater access to more evidence, facts, and observations—but fewer ways to build our trust in these, and even less time for deeper understanding. So, we are tempted to rely on technology to fill the trust gap.

Logical arguments are the extension of good evidence, facts, and observations. Some would say that it is an

DISCLAIMER

The authors are completely responsible for the content in this message. The opinions expressed here are their own.



activity that allows the mind to form conclusions in rigorous ways. When starting with facts and gathering evidence, logical arguments can lead to conclusions that are then based on facts and rigor. As this process is time consuming, we constantly decide whether the risk warrants the effort or whether to simply take the risk. Humans cut corners as part of our survival. Our intuition evolved through survival in nature. But now that things are changing, is our assessment of risk still serving us well?

CT is not natural for humans. We tend to use a process that functions in reverse; we make decisions and then form arguments to support the decision, often ignoring counterarguments and facts. We rationalize because we are not logical; we are instead rational. Sometimes it is our survival intuition that provides a ready solution for use when we must act; we use available additional time to test our solution, but usually, we search for evidence to help us be comfortable with the decision we already made. For survival reasons, we've been reinforced to not be critical thinkers!

As computer scientists and engineers, we may be familiar with the scientific method. The scientific method is based on logical arguments and follows a process of discovery involving asking questions, doing research, forming well-thought-out hypotheses, testing hypotheses, analyzing the results, and communicating those results clearly and correctly. By forming rigorous hypotheses that can be tested, facts can be supported. In this case, trust lies in the researcher. Placing trust in reported research often comes from research repeatability.

Skeptical is a term that is often used in this context—but sometimes incorrectly. It is not blind mistrust but rather the start of CT. But CT must be done next; otherwise, we are stuck in

denial. Thinking and reasoning allow movement and change. CT ensures that the movement is progress.

CT AND COMPUTERS

CT conflicts with computers in several ways. Logical arguments are not a new technology, but new technology is helping us get better at it. Computer scientists and mathematicians have used computer assistance to form

we can't fully understand, challenging their explainability. As we create and use these new technologies, we have a lot of work ahead to bridge the trust gap to the many use cases these technologies promise to address. CT, done efficiently, is more necessary than ever.

As we develop these new compute capabilities and build trust in some of these, we can potentially use these

CT is the process of using evidence, facts, observations, and logical arguments to make decisions, choices, or conclusions.

and check proofs for decades, but the range of computer-assisted proofs is expanding.¹ Using retrieval-augmented reasoning and using symbolic reasoning and knowledge graphs, computers have promise to assist even more effectively in reasoning and logic.² But before computers can assist with CT, we still need to get better at CT ourselves.

The early days of computing were simpler than today. Artificial intelligence (AI), generative AI (GAI), artificial general intelligence, and explainable AI (XAI) are more complex to verify and difficult to trust. XAI is a step in the right direction, for example, but most work to make AI explainable only helps to rationalize the output of AI from the inputs, not actually explaining the rationale or logic.

Fortunately, as developers of these platforms become more efficient, the testing tools are evolving to provide some insights into the closed boxes. Agentic AI, for example, helps us evolve agents that have a well-understood scope in logic, and that helps with explainability and therefore trust. But these complex systems of AI continue to behave in ways that

same tools for CT. Turning back to GAI, we already use multiple models to create question-and-answer pairs and then use another model to judge the answers from the first model and provide a rating of answer quality. But humans remain in that loop; we automate a lot—but only to scale the humans to address larger problems.

We can (and do) treat these software models as software but in a way that functions and is tested in a more human-like way. It is conceivable that we automate verification well enough that these models become more trustworthy than many humans, including experts in specific fields.

COMPUTING THE CRITICAL PATH

CT is taught—but not typically in engineering degree programs. Instead, it must be sought out and developed from multiple sources. Forensics and debate, philosophy, and fields of science provide a context that relies on logic and elements of CT but seldom covers the topic directly.

An easy place to start (and a worthy reminder) is to be familiar with and apply the logical fallacies to your reasoning; a good primer on structuring logical

arguments and avoiding pitfalls can be found in a few places, including “Logical Fallacies.”³ And once confident in the application of logical fallacies, be on guard for the Dunning-Kruger effect.⁴

So, is CT more critical today than before? Is it critical today at all? What do you think? We hope you do! ☐

REFERENCES

1. C. Drösser and G. Stix, “AI will become mathematicians’ co-pilot,” *Scientific American*, Jun. 8, 2024. Accessed: Mar. 18, 2025. [Online]. Available: <https://www.scientificamerican.com/article/ai-will-become-mathematicians-co-pilot/>
2. J. Duez, “The importance of logical reasoning in AI,” May 22, 2024. Accessed: Mar. 18, 2025. [Online]. Available: <https://www.forbes.com/councils/forbestechcouncil/2024/05/22/the-importance-of-logical-reasoning-in-ai/>
3. “Logical fallacies.” The Skeptics Guide to the Universe. Accessed: Mar. 18, 2025. [Online]. Available: <https://legacy.theskepticsguide.org/resources/logical-fallacies>
4. “Dunning-Kruger effect.” Psychology Today. Accessed: Mar. 18, 2025. [Online]. Available: <https://www.psychologytoday.com/us/basics/dunning-kruger-effect>

JASON RUPE is a distinguished technologist at Cable Labs, Louisville, CO 80027 USA, and is the president of the IEEE Reliability Society. Contact him at jrupe@ieee.org.

JEFFREY VOAS, Gaithersburg, MD 20899 USA, is the editor in chief of *Computer*. He is a Fellow of IEEE. Contact him at j.voas@ieee.org.

Unlock Your Potential

WORLD-CLASS CONFERENCES — Over 195 globally recognized conferences.

DIGITAL LIBRARY — Over 900k articles covering world-class peer-reviewed content.

CALLS FOR PAPERS — Write and present your ground-breaking accomplishments.

EDUCATION — Strengthen your resume with the IEEE Computer Society Course Catalog.

ADVANCE YOUR CAREER — Search new positions in the IEEE Computer Society Jobs Board.

NETWORK — Make connections in local Region, Section, and Chapter activities.



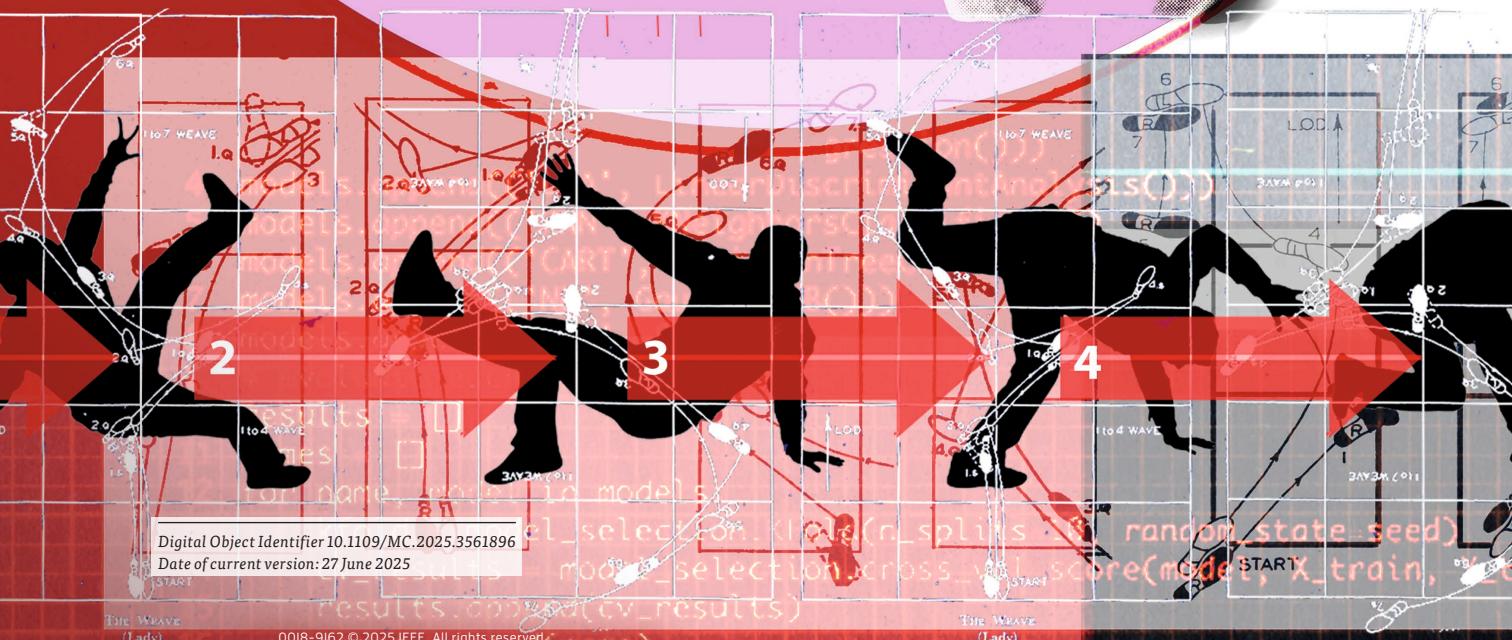
Explore membership today
at the IEEE Computer Society
www.computer.org



Generative AI in the Software Development Lifecycle

Tracy (Trac) Bannon^{ID}, MITRE Corporation

Phil Laplante^{ID}, IEEE Fellow



This special issue is a thoughtful exploration of how generative AI is reshaping software engineering—highlighting the balance between automation and human judgment across the modern software development lifecycle (SDLC).

Just because we can automate something doesn't mean we should. There, we said it. Automate what's repeatable. Let humans drive what's strategic. That's the guiding principle behind our exploration of generative artificial intelligence's (AI's) role in software engineering and much of what is borne out by the papers offered by the contributors. As guest editors of this special issue, we've seen firsthand how AI is both accelerating development and raising important questions. Yes, large language models (LLMs) can assist developers by automating code generation and assisting with bug fixes, but they also risk eroding foundational skills like problem decomposition, architectural reasoning, and critical thinking.¹

In DevSecOps and modern software practices, we love automation. It can speed up deployments, improve consistency, and help us scale. But we need to be clear about where automation shines—and where human cognition is still irreplaceable.

DISCLAIMER

The author's affiliation with The MITRE Corporation is provided for identification purposes only and is not intended to convey or imply MITRE's concurrence with, or support for, the positions, opinions, or viewpoints expressed by the author.

The temptation to automate everything is real—especially given the myriad industry tool reports. But researchers and practitioners need to cut through the hype and focus on data-driven insights. We must recognize that not all work fits into a checklist. When architects and engineers design architectures, assess risk, or troubleshoot failures, we aren't following a linear script. We are synthesizing information, weighing tradeoffs, and making decisions in complex, ever-changing environments. Yet, as DevSecOps has taught us, it's not just about speed but about doing it right. Automating too much too fast can lead to skills atrophy, collaboration breakdowns, technical debt, and even code vulnerabilities. A recent Microsoft study even suggests that the more users trust AI, the less they challenge it—raising the stakes for thoughtful integration.²

One technique that helps us reframe this balance is goal-directed task analysis (GDTA). Instead of breaking work into predefined steps, GDTA focuses on three elements: what goals engineers must achieve, what decisions need to be made, and what information is necessary to make those decisions effectively.³ Applying this thinking to DevSecOps and modern software practices changes the way we think about human involvement. We must care about the value stream, remove friction, and focus on the humans in the loop today and humans on the loop in the future.

Our challenge to the community: balance innovation with intentionality, speed with sustainability, and delegation with discernment. Let's embrace the possibilities without losing the human touch that makes for great software engineering. After all, as we were reminded of a recent story of Italian mafia godfathers

ABOUT THE AUTHORS

TRACY (TRAC) BANNON is senior principal, software architect, and researcher with the MITRE Corporation, Bedford, MA 01730 USA. Contact her at tbannon@mitre.org.

PHIL LAPLANTE, State College, PA 16801 USA, is a computer scientist and software engineer, a Fellow of IEEE, and an associate editor in chief of *Computer*. Contact him at plaplante@psu.edu.

lamenting the lost skills of the younger generation—where even the basics of wielding a gun or committing a robbery are being forgotten⁴—it's a cautionary tale: When skills aren't practiced, they fade. "Leave the gun, take the cannoli"—this reminder shouldn't even need to be given, but some skills, even the most obvious, must be taught and practiced. We're certainly not comparing all software engineers to mobsters, but the point holds—let's not lose our foundational skills while embracing the new.

We hope this special issue inspires both reflection and action. Dive in and see how AI is transforming our field—strategically, thoughtfully, and with a bit of humor along the way. □

REFERENCES

1. T. T. Bannon, "Infusing artificial intelligence into software engineering and the DevSecOps continuum," *Computer*, vol. 57, no. 9, pp. 140–148, Sep. 2024, doi: [10.1109/MC.2024.3423108](https://doi.org/10.1109/MC.2024.3423108).
2. E. Shein, "The impact of AI on computer science education," *Commun. ACM*, vol. 67, no. 9, pp. 13–15, Jun. 2024, doi: [10.1145/3673428](https://doi.org/10.1145/3673428).
3. M. R. Endsley, *Designing for Situation Awareness: An Approach to User-Centered Design*, 2nd ed. Boca Raton, FL, USA: CRC Press, 2011.
4. N. Squires, "Mafia godfathers lament 'miserable' calibre of new recruits," *The Telegraph*, Feb. 11, 2025. Accessed: Mar. 25, 2025. [Online]. Available: <https://www.yahoo.com/news/wiretaps-reveal-miserable-state-italian-l74338001.html>

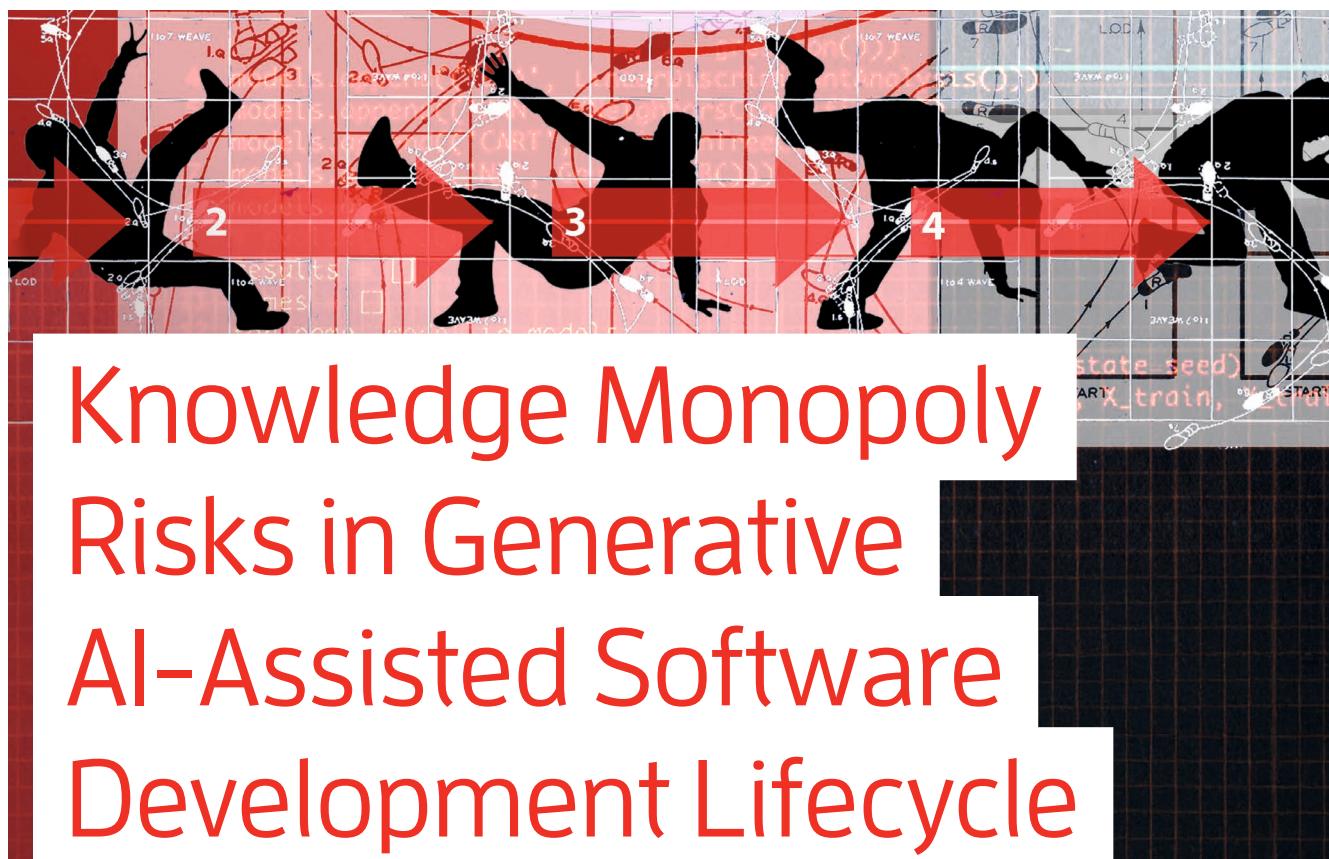
SUBMIT
TODAY

IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING

► SUBSCRIBE AND SUBMIT

For more information on paper submission, featured articles, calls for papers, and subscription links visit: www.computer.org/tsusc





Knowledge Monopoly Risks in Generative AI-Assisted Software Development Lifecycle

Yanyi Wu^{ID}, Xinyu Lu, and Chenghua Lin^{ID}, Zhejiang University

This article examines the emerging knowledge monopolization risks in generative artificial intelligence (AI)-assisted software development, proposing a comprehensive assessment framework and practical intervention strategies to help organizations maintain technical sovereignty while leveraging AI capabilities for development efficiency.

Imagine a software developer's surprise when their artificial intelligence (AI) assistant starts physically controlling their computer: moving the cursor, scrolling through documents, and completing forms autonomously. This scenario became a reality in October 2024 when Anthropic unveiled Claude 3.5 Sonnet, showcasing unprecedented capabilities in computer interaction and development assistance. Such advancement marks more than just another milestone in AI evolution; it signals a fundamental shift in how software

development knowledge is created, accessed, and controlled. As AI systems transition from passive code completion tools to active development participants, questions arise about the future of human expertise and technical knowledge sovereignty in software teams.

Across development teams worldwide, generative AI has already reshaped traditional software development practices. Stack Overflow's 2024 Developer Survey reveals striking statistics: 76% of all respondents are using or are planning to use AI tools in their development process this year, an increase from last year (70%).¹ AI adoption now permeates all aspects of development, from simple coding assistance through architectural

Digital Object Identifier 10.1109/MC.2025.3532947
Date of current version: 27 June 2025

decisions, to debugging strategies and project planning. Traditional knowledge-building practices, such as peer code reviews and documentation writing, face subtle yet significant transformations. While AI tools promise efficiency and rapid solution generation, they simultaneously alter the fundamental dynamics of how development teams learn, collaborate, and maintain their technical expertise.

Knowledge monopoly in AI-assisted software development manifests in ways far more nuanced than simple tool dependency. Development teams find themselves at a crossroads: embracing AI's capabilities while potentially surrendering their role as primary knowledge creators. Consider a junior developer who routinely solves complex problems through AI prompts rather than developing deep understanding through traditional learning paths. Or a senior architect who increasingly defers to AI recommendations for system design decisions. These scenarios illustrate a shifting paradigm where technical knowledge, once primarily human-centered and collectively built, gradually concentrates within AI systems. Modern development challenges transcend mere efficiency gains and productivity metrics, fundamentally reshaping sustainable software practices and team innovation capabilities.² As AI tools grow more sophisticated, the boundary between augmentation and dependence becomes increasingly blurred, demanding careful consideration of how teams can maintain their knowledge sovereignty while leveraging AI's powerful capabilities.

THE HIDDEN COST OF GENERATIVE AI ASSISTANCE

AI development tools' evolution marks a fundamental transformation

in software engineering practices, challenging traditional paradigms of technical knowledge acquisition and retention. Modern AI assistants, exemplified by GitHub Copilot and Amazon CodeWhisperer, have transcended simple code completion to become sophisticated systems capable of generating entire functions, debugging complex systems, and proposing architectural solutions. Gartner estimates that by 2026, more than 80% of companies are expected to utilize generative AI application programming interfaces or models, or deploy generative AI-powered applications in their operations, marking a significant rise from the less than 5% observed in 2023.³ Widespread adoption has initiated a subtle but profound shift in how technical knowledge is created, shared, and maintained within organizations.

Technical competencies are eroding most visibly in routine development tasks, giving rise to a "competency paradox." Despite their increased productivity in code generation, developers' fundamental problem-solving capabilities show concerning patterns of decline. Such transformation is particularly evident in their approach to technical challenges. Traditional problem-solving approaches have been largely replaced by an emphasis on optimizing AI interactions, shifting focus away from understanding core technical concepts. McKinsey's longitudinal study of developer behavior patterns reveals that teams heavily reliant on AI assistance spend 10% less time on high-complexity tasks compared to their counterparts who maintain a balanced approach.⁴

Beyond routine coding, this phenomenon has profoundly affected debugging and system understanding. Traditional debugging processes—reading

stack traces, stepping through code, and forming hypotheses about program behavior—increasingly give way to AI-assisted approaches where developers often request solutions without comprehending root causes. Gradually emerging is the "black box developer" syndrome, where engineers solve technical problems without developing the mental models essential for long-term expertise. The phenomenon becomes particularly problematic in distributed systems and microservices architectures, where understanding system behavior requires deep technical insight rather than pattern matching.⁵

A more subtle but equally concerning trend emerges in specialized technical domains. In machine learning engineering, for instance, developers increasingly rely on AI to optimize neural network architectures and tune hyperparameters. Growing dependency leads to "algorithmic alienation," a state where developers become disconnected from the fundamental principles of their work. High task familiarity leads developers to become overconfident in their own ability, yet research shows the more confident users are, the less well they perform when working with AI systems, particularly when dealing with out-of-distribution data where performance is uncertain.⁶

Security implications present another critical dimension of this transformation. The heavy reliance on AI-generated code can introduce potential security vulnerabilities, particularly in areas requiring deep understandings of security principles.⁷ When developers overly depend on AI for code generation without fully comprehending the security implications, subtle vulnerabilities can emerge in critical areas, such as authentication and data protection. This risk becomes

more significant when considering how these vulnerabilities might propagate across multiple projects through shared AI models. The situation grows more complex when examining long-term code maintenance and technical debt. Development teams increasingly

where junior developers gradually build expertise through hands-on experience and mentorship, faces disruption from AI tools that provide immediate solutions without corresponding understanding. Across the industry, a concerning “expertise gap”

many now engage in “prompt engineering”: crafting the perfect AI query rather than understanding the underlying technical challenge. The transformation fundamentally alters the learning patterns that traditionally built deep technical expertise, potentially creating gaps in fundamental understanding that become evident in complex problem-solving scenarios.⁹

Debugging capabilities suffer equally, if not more severely, than basic coding skills. The traditional debugging process—reading stack traces, stepping through code, and forming hypotheses about program behavior—gives way to an AI-assisted approach where developers often request solutions without fully comprehending the root cause. This pattern has evolved into a concerning trend where developers solve complex problems without developing the fundamental understanding necessary for sustainable expertise.¹⁰ The diminishing ability to independently troubleshoot complex system issues becomes particularly apparent in distributed systems and concurrent programming scenarios, where understanding system behavior requires deep technical insight rather than pattern matching.

Perhaps most concerning is the impact on code optimization capabilities. While AI tools excel at generating functionally correct code, the nuanced understanding of performance implications often eludes both the AI and its users.¹¹ Development teams report a growing tendency to accept default implementations without considering memory usage, processing efficiency, or scalability concerns. This trend is particularly evident in resource-constrained environments, where performance optimization traditionally required deep technical knowledge and careful consideration of tradeoffs.

ACROSS THE INDUSTRY, A CONCERNING “EXPERTISE GAP” HAS SURFACED, WHERE SURFACE-LEVEL PROBLEM-SOLVING ABILITIES MASK A DEEPER LACK OF TECHNICAL COMPREHENSION.

face the “AI dependency cycle,” where maintaining AI-generated code demands ever more AI assistance, further widening the gap in deep system understanding.

Knowledge sharing and team dynamics have undergone radical shifts in response to these changes. Traditional mechanisms of technical knowledge transfer—peer reviews, architecture discussions, and mentorship programs—show signs of deterioration when teams over-rely on AI assistance.⁸ The change affects not just immediate problem-solving capabilities but the very foundation of how technical expertise develops within organizations. The phenomenon mirrors historical patterns of technological adoption, but with potentially more severe implications, given software development’s central role in modern technological infrastructure.

Most critically, emerging trends have upended fundamental assumptions about technical expertise development. The traditional apprenticeship model of software development,

has surfaced, where surface-level problem-solving abilities mask a deeper lack of technical comprehension.

KNOWLEDGE MONOPOLY PATTERNS

Software teams embracing AI tools find themselves navigating an intricate landscape where traditional knowledge dynamics are rapidly shifting. As AI capabilities expand, the patterns of knowledge creation, retention, and application undergo subtle yet profound transformations.

Technical competency impact

Across development teams, declining technical competencies surface most prominently during everyday coding activities. Across the industry, the delegation of programming challenges to AI assistants—from basic algorithm implementation to complex data structure manipulation—has fundamentally altered traditional problem-solving approaches. Where developers once methodically decomposed problems and explored multiple solutions,

The ability to effectively profile and optimize applications becomes increasingly rare as developers rely more heavily on AI-generated solutions without developing the underlying performance engineering expertise.

The shift in problem-solving patterns suggests a deeper transformation in how technical expertise develops. Traditional learning pathways, built on incremental understanding and hands-on experience, are being shortened or bypassed entirely. While AI tools provide immediate solutions, they may inadvertently interrupt the essential cognitive processes that build robust technical intuition. This creates a paradoxical situation where code production efficiency increases while deep technical understanding potentially decreases. The impact becomes particularly visible in architectural decisions and system design, where success depends more on comprehensive technical judgment than on individual coding solutions.

Architecture and design intelligence

Software architecture perhaps reveals the most concerning pattern of knowledge monopolization. As AI systems become more adept at suggesting architectural patterns and design solutions, the nuanced understanding of tradeoffs and context-specific considerations begins to fade. Particularly concerning is development teams' growing tendency to accept AI-recommended architectural decisions without the once-standard rigorous evaluation processes.¹² Reliance on AI-suggested solutions has gradually superseded innovative approaches tailored to unique business requirements, resulting in diminished innovation capabilities. The erosion of

architectural reasoning capabilities becomes especially apparent when teams face unprecedented technical challenges that demand original thinking rather than pattern matching.

Declining quality of architectural discussions reflects this troubling shift. Where design meetings once featured robust debates about competing approaches, many teams now default to AI suggestions, reducing the collective learning that emerges from diverse perspectives.¹³ The depth and breadth of architectural exploration diminish as teams increasingly rely on AI to provide "proven" solutions. Novel requirements and unique system constraints particularly highlight the challenges of current development approaches. In addition to affecting immediate decision making, the reduction in collaborative architectural discourse also impairs the development of shared architectural knowledge within teams. The traditional process of building collective wisdom through rigorous debate and diverse viewpoints gives way to more passive acceptance of AI-generated recommendations.

What's more, system architecture's ability to evolve and adapt has significantly deteriorated over time. Teams increasingly struggle to make informed decisions about architectural refactoring, often relying on AI to suggest modernization approaches without fully understanding the technical debt implications.¹⁴ Software teams increasingly face "architectural brittleness," where systems become difficult to evolve because teams lack the deep architectural knowledge required to make substantive changes. Immediate architectural decisions cascade into profound implications for system maintainability and long-term evolution. Teams demonstrate

reduced capability in identifying architectural decay patterns and planning strategic refactoring initiatives, particularly when facing complex legacy system modernization.

The impact on architectural innovation deserves special attention. Creative tension between technical constraints and business requirements once drove traditional architectural innovation, compelling teams to discover unconventional solutions. However, the readily available AI-suggested solutions might inadvertently discourage this type of innovative thinking. Teams show a decreasing tendency to explore unconventional architectural approaches or challenge established patterns, even when traditional solutions may not optimally serve their specific context. Subtle patterns of knowledge erosion emerge in developers' declining ability to think architecturally outside of established patterns and solutions.

Assessing knowledge monopoly risks

The patterns of technical competency erosion and architectural knowledge loss discussed above point to a critical need: Organizations must systematically evaluate their vulnerability to AI-induced knowledge monopolization. To address this need, we present a structured framework for assessing knowledge monopoly risks across different phases of software development. This framework aims to help organizations identify potential knowledge gaps before they become critical impediments to sustainable development practices.

Our framework builds upon three fundamental dimensions that emerged from the previous analysis. Technical skills impact captures the degree to

APPLYING AI TO SOFTWARE DEVELOPMENT

which core development capabilities and deep technical understanding may be compromised by AI dependency. This dimension examines not just the ability to write code, but the deeper understanding of algorithms, system behaviors, and performance implications. *Knowledge transfer risk* examines the potential barriers to effective expertise sharing and knowledge preservation within teams, which includes both explicit knowledge captured in documentation and implicit knowledge traditionally shared through mentorship and collaboration. *Innovation capability impact* evaluates how AI tools might affect a team's ability to devise original solutions and adapt to novel challenges, particularly in scenarios where existing patterns and solutions prove insufficient.

The framework provides a systematic approach for organizations to evaluate their development practices through these three lenses. Each dimension can be assessed using concrete indicators specific to different

development phases. For example, when evaluating technical skills impact, organizations might examine developers' ability to perform independent problem decomposition, understand algorithmic complexity, or conduct thorough root cause analysis. Similarly, knowledge transfer risk assessment might focus on the quality of technical documentation, the effectiveness of peer learning processes, and the preservation of decision rationales. These assessments should be conducted periodically to track changes in knowledge dynamics over time.

Organizations can implement this framework through structured review sessions, developer surveys, and performance metrics analysis. The assessment process should involve both senior and junior team members to capture different perspectives on knowledge flow within the organization. Regular evaluation helps identify trends and patterns in how AI tool usage affects different aspects of technical knowledge management. For instance,

teams might discover that while AI tools accelerate development in certain areas, they may simultaneously create blind spots in system understanding or architectural reasoning.

The practical application of this framework is designed to be flexible yet comprehensive. Organizations can adapt the assessment criteria based on their specific context, development methodologies, and technological stack, while maintaining the fundamental evaluation structure. It enables development teams to identify potential knowledge monopolization risks before they manifest as critical issues, allowing for proactive intervention in vulnerable areas. Teams can use the framework to develop targeted strategies for preserving crucial technical knowledge while still leveraging the benefits of AI assistance. **Table 1** presents the detailed assessment criteria across different phases of the software development lifecycle, providing a practical tool for organizations to begin their risk evaluation process.

TABLE 1. Knowledge monopoly risk assessment framework.

SDLC phase	Technical skills	Knowledge transfer	Innovation capability
Planning	<ul style="list-style-type: none">• Problem decomposition ability• Domain knowledge retention	<ul style="list-style-type: none">• Requirements documentation quality• Stakeholder communication patterns	<ul style="list-style-type: none">• Novel requirement interpretation• Solution ideation diversity
Design	<ul style="list-style-type: none">• Design pattern understanding• Tradeoff analysis capability	<ul style="list-style-type: none">• Architecture decision documentation• Design knowledge sharing practices	<ul style="list-style-type: none">• Novel architecture approaches• Architectural adaptation ability
Implementation	<ul style="list-style-type: none">• Algorithm comprehension• Debugging capability	<ul style="list-style-type: none">• Code documentation practices• Peer code review effectiveness	<ul style="list-style-type: none">• Implementation creativity• Technical solution diversity
Testing	<ul style="list-style-type: none">• Test strategy development• Edge case identification	<ul style="list-style-type: none">• Test knowledge documentation• Testing practice sharing	<ul style="list-style-type: none">• Test scenario innovation• Test approach adaptation
Deployment	<ul style="list-style-type: none">• Infrastructure understanding• Problem diagnosis ability	<ul style="list-style-type: none">• Deployment documentation• Operational knowledge sharing	<ul style="list-style-type: none">• Deployment strategy innovation• Pipeline optimization capability
Maintenance	<ul style="list-style-type: none">• System comprehension level• Root cause analysis ability	<ul style="list-style-type: none">• Maintenance knowledge retention• Historical context preservation	<ul style="list-style-type: none">• Enhancement innovation• System evolution capability

BALANCING AI INTEGRATION AND KNOWLEDGE PRESERVATION

Having identified the patterns and risks of knowledge monopolization, organizations face the challenge of maintaining their technical sovereignty while benefiting from AI assistance. The balance between leveraging AI capabilities and preserving deep technical knowledge requires careful consideration and systematic intervention.

Risk-driven intervention approaches

To be truly effective, intervention strategies must align closely with specific organizational contexts and risk patterns. Rather than implementing blanket restrictions on AI usage, organizations need a carefully calibrated approach that targets identified risk areas while maintaining productivity. [Table 2](#) presents a comprehensive strategy selection matrix to guide organizations in choosing and implementing appropriate interventions

based on different risk scenarios and organizational contexts.

Drawing from this strategic matrix, organizations can implement specific interventions for different risk scenarios. In high-risk areas, especially those with elevated technical skills impact scores, “progressive resistance training” has proven invaluable in technical skills development, introducing increasingly complex challenges beyond the reach of AI tools. Teams begin with algorithm optimization challenges where developers must improve existing solutions for better performance or memory usage.¹⁵ These challenges evolve into distributed system design problems that require an understanding of consistency models, network partitioning, and fault tolerance. The final stage involves novel architectural problems addressing unique business requirements that haven’t been widely solved in the industry. Central to success is the mandatory code review process where developers must explain their solutions’

performance characteristics, scaling properties, and potential failure modes.

Successfully preserving knowledge transfer mechanisms demands a sophisticated approach transcending traditional documentation practices. Organizations successful in this area implement a three-tier knowledge preservation system.¹⁶ At the foundational level, teams maintain “technical decision journals” that capture not just final decisions but the complete decision-making process. These journals document alternative approaches considered, performance implications, scalability concerns, and maintenance tradeoffs. The second tier involves weekly “architectural reflection sessions” where teams analyze recent technical decisions and their potential long-term impacts. The highest tier consists of monthly “system evolution workshops,” where developers collaborate to predict future scaling challenges and design potential solutions. The tiered approach ensures that knowledge transfer becomes an

TABLE 2. Risk-driven intervention strategy selection.

Risk scenario	Primary strategy	Supporting activities	Resource requirements	Success indicators
High technical skills impact	AI-free development sessions	<ul style="list-style-type: none">Algorithmic problem-solving workshopsCode review deep divesSystem design challenges	<ul style="list-style-type: none">Senior developer timeTraining materialsAssessment tools	<ul style="list-style-type: none">Independent problem-solving metricsTechnical assessment scoresCode quality metrics
Elevated knowledge transfer risk	Enhanced mentorship program	<ul style="list-style-type: none">Pair programming sessionsDocumentation sprintsKnowledge sharing forums	<ul style="list-style-type: none">Mentorship timeDocumentation platformCollaboration tools	<ul style="list-style-type: none">Knowledge transfer ratesDocumentation qualityTeam collaboration metrics
Declining innovation capability	Architecture Katas program	<ul style="list-style-type: none">Innovation workshopsSystem evolution planningNovel problem challenges	<ul style="list-style-type: none">Workshop facilitatorsDesign toolsReference materials	<ul style="list-style-type: none">Novel solution generationArchitectural innovation metricsProblem-solving diversity
Mixed risk profile	Hybrid intervention approach	<ul style="list-style-type: none">Balanced AI usage guidelinesSelective tool applicationRegular skill assessments	<ul style="list-style-type: none">Mixed resource poolAssessment frameworkMonitoring tools	<ul style="list-style-type: none">Balanced performance metricsTeam growth indicatorsTool usage optimization

APPLYING AI TO SOFTWARE DEVELOPMENT

integral part of the development process rather than an afterthought.

Innovation capability preservation demands a structured yet flexible framework that actively cultivates creative problem-solving skills. Organizations implement “constraint programming sessions” with specific technical limitations that prevent direct AI assistance. For example, teams might need to optimize a system’s memory usage while maintaining its performance characteristics, or redesign a service to handle a 10× increase in load without adding hardware resources. These sessions follow a structured format: problem presentation, individual solution development, peer review, and collective refinement. The key innovation here lies in the requirement for multiple viable solutions rather than a single optimal approach. Teams must present at least three different architectures for each significant technical challenge, analyzing the tradeoffs and context-specific advantages of each approach.

Continuous assessment and adaptation form the cornerstone of successful intervention strategies. Organizations implement a quantitative evaluation framework that tracks multiple dimensions of team performance. Technical competency metrics measure developers’ ability to solve problems independently, understand system behaviors, and make informed architectural decisions. Knowledge transfer effectiveness is evaluated through peer review quality, documentation comprehensiveness, and the success rate of knowledge application in new contexts. Innovation capabilities are assessed through the diversity of solution approaches, the quality of architectural decisions, and the team’s ability to adapt existing patterns to novel challenges. Its comprehensive

measurement system enables organizations to fine-tune their intervention strategies based on objective data rather than subjective impressions.

Modern development environments require an additional layer of consideration: the integration of these practices with existing development workflows. Leading teams have embraced “AI-augmented learning cycles”: structured periods where AI serves not as a solution provider, but as a catalyst for accelerated learning. During these cycles, developers first attempt problems independently, then use AI tools to explore alternative approaches and finally analyze the differences between human-generated and AI-generated solutions. The comparative analysis helps teams understand both the strengths and limitations of AI assistance, while maintaining their core technical capabilities.

Practical knowledge retention mechanisms

Knowledge retention in AI-augmented development environments requires mechanisms that go beyond traditional documentation practices. Based on the risk assessment results from our framework, we propose a set of practical mechanisms that specifically target different aspects of knowledge preservation. [Table 3](#) provides these knowledge retention mechanisms and their implementation approaches. These mechanisms need to be both systematic in their approach and flexible in their implementation to accommodate various team structures and development contexts. The challenge lies not just in capturing knowledge, but in maintaining its accessibility and relevance over time.

At the core of effective knowledge retention lies the concept of “technical

context preservation chains.” Its mechanism operates at three levels: code, architecture, and system evolution. At the code level, teams implement “explanation checkpoints” where developers must articulate the reasoning behind complex implementations before committing code. These checkpoints focus not on what the code does, but on why specific approaches were chosen over alternatives. For instance, developers must explain performance tradeoffs, memory considerations, and maintainability implications.¹⁷ At the architecture level, teams maintain “design decision trees” that capture not only the chosen solutions but also the explored alternatives and their tradeoffs. Each node in these trees represents a design decision point, complete with context, constraints, and the reasoning behind the final choice. The system evolution level involves regular “architectural storytelling sessions” where teams construct narratives about how their systems evolved, including failed approaches and learned lessons. These sessions are recorded and indexed, creating a valuable knowledge repository that captures the human expertise behind technical decisions.

A critical component of knowledge retention is the preservation of problem-solving patterns. Organizations implement “solution pattern libraries” that document not just successful solutions, but the thought processes leading to them. These libraries differ from traditional pattern repositories in their focus on capturing the cognitive journey of problem-solving. Teams maintain structured entries that include initial problem framing, solution exploration strategies, and decision rationales. Each entry captures multiple dimensions: technical constraints,

business requirements, performance considerations, and scalability implications. The pattern libraries also include a specialized section called “AI–Human Solution Analysis,” where teams document cases where AI-suggested solutions were modified or rejected in favor of human-designed alternatives. The section is particularly valuable for preserving critical thinking skills and understanding the limitations of AI-generated solutions.

When seamlessly woven into daily development practices, these mechanisms demonstrate their highest value and impact. The most resilient teams have integrated “knowledge activation points” throughout their development process, creating natural triggers for knowledge capture and transfer. These include preimplementation design reviews, where teams must present multiple solution approaches, post-implementation retrospectives that analyze both successful and unsuccessful technical decisions, and regular system evolution workshops that

explore future scaling challenges.¹⁸ Teams also conduct bi-weekly “technical debt awareness sessions,” where they examine how current decisions might impact future system evolution. These sessions help preserve the crucial skill of technical foresight that AI tools cannot readily provide.

Knowledge retention mechanisms must also address the challenge of maintaining technical intuition. Organizations implement “intuition building workshops,” where developers tackle complex problems without immediate access to solutions. These workshops cultivate “technical pattern recognition” abilities, enabling developers to instinctively spot potential issues and opportunities in system design. Teams practice scenarios, such as performance optimization, scalability planning, and failure mode analysis. The workshops often include “system behavior prediction exercises,” where developers must anticipate how systems will respond to various stress conditions before running actual tests.

The practical implementation of these mechanisms requires careful attention to team dynamics and organizational culture. Successful organizations establish “knowledge sharing incentive systems” that recognize and reward effective knowledge transfer. These systems might include technical mentorship programs, where experienced developers are explicitly tasked with preserving and transmitting critical knowledge. Teams also implement “cross-pollination sessions,” where developers from different projects share their experiences and insights, helping to build a broader understanding of system design principles across the organization.

RETHINKING SOFTWARE ENGINEERING

AI-augmented development has fundamentally reshaped our understanding of software engineering practices. The assessment framework and mitigation strategies proposed in this article represent initial steps toward

TABLE 3. Knowledge retention mechanisms.

Knowledge type	Retention mechanism	Implementation approach	Key activities	Success metrics
Technical context	Technical context preservation chains	<ul style="list-style-type: none"> Code level Architecture level System evolution level 	<ul style="list-style-type: none"> Explanation checkpoints Design decision trees Architectural storytelling 	<ul style="list-style-type: none"> Documentation quality Decision tree completeness Knowledge retrieval success rate
Problem-solving patterns	Solution pattern libraries	<ul style="list-style-type: none"> Pattern documentation Cognitive journey mapping AI–human solution analysis 	<ul style="list-style-type: none"> Problem framing sessions Solution exploration workshops Tradeoff analysis 	<ul style="list-style-type: none"> Pattern reuse rate Solution quality Team problem-solving speed
Technical intuition	Intuition building workshops	<ul style="list-style-type: none"> Pattern recognition training System behavior prediction Performance analysis 	<ul style="list-style-type: none"> Prediction exercises System stress testing Failure mode analysis 	<ul style="list-style-type: none"> Prediction accuracy Response time to issues System understanding depth
Organizational knowledge	Cross-pollination programs	<ul style="list-style-type: none"> Mentorship programs Knowledge sharing incentives Team rotation 	<ul style="list-style-type: none"> Cross-team sessions Technical presentations Knowledge sharing events 	<ul style="list-style-type: none"> Knowledge transfer rate Team collaboration metrics Innovation indicators

APPLYING AI TO SOFTWARE DEVELOPMENT

addressing knowledge monopoly risks, but the implications run deeper than technical solutions alone. The transformation we're witnessing isn't merely about tool adoption: It's about the changing nature of technical knowledge creation and preservation. Traditional knowledge management approaches, built around documentation and mentorship, struggle to capture the rapid evolution of AI-assisted development practices.¹⁹ Moreover, the risk of knowledge monopolization brings forth questions about the very essence of technical expertise and how it should be cultivated in modern development environments.

The relationship between technical capabilities and knowledge inheritance is undergoing a profound transformation. As AI tools continue to excel at generating solutions, the traditional learning journey that helped developers build deep technical intuition becomes increasingly obscured. Consequently, teams face unprecedented challenges in acquiring and maintaining expertise. The cognitive patterns of problem-solving are evolving; developers increasingly need to master the art of problem framing and solution evaluation, rather than just solution generation. Emerging technical landscapes demand a new understanding of competency, emphasizing critical evaluation and adaptation of AI-generated solutions rather than creating them from scratch. Mastery of "metatechnical skills"—the ability to navigate and leverage AI capabilities while maintaining independent technical judgment—has become essential in today's development landscape.²⁰

Immediate architectural decisions cascade into profound implications for system maintainability and long-term evolution. Development teams are

discovering that their traditional structures and collaboration patterns may not effectively support knowledge preservation in AI-augmented environments. The classical mentor-mentee relationship, for instance, needs reinvention when AI tools can provide instant solutions that historically required years of experience to develop. Modern organizations must cultivate a "collective technical consciousness": fostering a shared understanding of system behaviors and architectural principles that transcends individual expertise. Collective knowledge becomes crucial as AI tools democratize code generation but potentially centralize deeper technical understanding.²¹ Teams successful in this transition have developed new forms of technical dialogue that explicitly address the gap between AI-generated solutions and human-centered system comprehension.

Striking an optimal balance between efficiency gains and knowledge preservation continues to challenge development organizations. We suggest that the solution lies not in restricting AI tool usage, but in fundamentally reimagining how technical knowledge is created, shared, and maintained. The concept of technical autonomy needs redefinition in an era where AI can generate sophisticated solutions instantly. Redefining technical capability demands recognition of comprehensive competencies: not just code generation, but deep system understanding, architectural reasoning, and innovation potential. Organizations must now focus on maintaining technical skills and cultivating "adaptive technical intelligence": evolving technical knowledge alongside AI tools rather than competing against them.

Looking forward, the software engineering discipline faces a potential

paradigm shift. The traditional emphasis on coding skills and design patterns may need to evolve toward a greater focus on system thinking and architectural reasoning. The future software engineer might be less a code creator and more a technical strategist, combining deep system understanding with effective AI tool orchestration.²² Software engineering education and professional development need to embrace a new model, prioritizing uniquely human strengths: creativity, critical thinking, and strategic technical decision making. The key to maintaining technical sovereignty in an AI-augmented future may lie in recognizing and nurturing these distinctly human capabilities while leveraging AI tools to enhance rather than replace human technical judgment.

Knowledge monopolization risk in software engineering unveils a deeper challenge: the fundamental shift in how human technical expertise is developed and maintained in AI-augmented environments. The assessment framework and mitigation strategies presented mark only initial steps toward addressing this transformation. Software engineering must evolve toward sophisticated forms of human-AI collaboration, preserving essential engineering judgment while embracing AI capabilities. Rather than viewing AI tools as potential threats, the focus should shift toward leveraging these tools to enhance human capabilities in ways that maintain technical sovereignty. Future research must explore adaptive learning mechanisms that ensure technical innovation remains driven by human insight rather than constrained by tool limitations. □

ACKNOWLEDGMENT

This research was supported by the General Project of the National Social Science Fund of China under Grant BIA230181.

REFERENCES

1. Stack Overflow. "2024 stack overflow developer survey." Stack Overflow Annual Developer Survey. [Online]. Available: <https://survey.stackoverflow.co/2024/>
2. T. Birkstedt, M. Minkkinen, A. Tandon, and M. Mäntymäki, "AI governance: Themes, knowledge gaps and future agendas," *Internet Res.*, vol. 33, no. 7, pp. 133–167, Jun. 2023, doi: [10.1108/INTR-01-2022-0042](https://doi.org/10.1108/INTR-01-2022-0042).
3. "Gartner says more than 80% of enterprises will have used generative AI APIs or deployed generative AI-enabled applications by 2026," *Gartner*, Oct. 11, 2023. [Online]. Available: <https://www.gartner.com/en/newsroom/press-releases/2023-10-11-gartner-says-more-than-80-percent-of-enterprises-will-have-used-generative-ai-apis-or-deployed-generative-ai-enabled-applications-by-2026>
4. "Unleash developer productivity with generative AI." McKinsey. [Online]. Available: <https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/unleashing-developer-productivity-with-generative-ai#/>
5. A. Vomberg, E. de Haan, N. E. Fabian, and T. Broekhuizen, "Digital knowledge engineering for strategy development," *J. Bus. Res.*, vol. 177, Apr. 2024, Art. no. 114632, doi: [10.1016/j.jbusres.2024.114632](https://doi.org/10.1016/j.jbusres.2024.114632).
6. S. Passi and M. Vorvoreanu, "Over-reliance on AI: Literature review." Microsoft. [Online]. Available: <https://www.microsoft.com/en-us/research/publication/overreliance-on-ai-literature-review/>

ABOUT THE AUTHORS

YANYI WU is a research associate with the School of Public Affairs and the Institute of China's Science, Technology and Education Policy, Zhejiang University, 310058 Hangzhou, China. His research interests include the governance of emerging technologies. Contact him at yanyi.wu@hotmail.com.

XINYU LU is a Ph.D. student with the School of Public Affairs and the Institute of China's Science, Technology and Education Policy, Zhejiang University, 310058 Hangzhou, China. Her research interests include the social impacts of new tech, policy-driven innovation diffusion, and the ethics of emerging scientific breakthroughs. Lu received an M.A. in economics from Capital University of Economics and Business. Contact her at 12422079@zju.edu.cn.

CHENGHUA LIN is a professor with the School of Public Affairs and the Institute of China's Science, Technology, and Education Policy, Zhejiang University, 310058 Hangzhou, China. His research interests include interdisciplinary research, especially on the integration of science and technology innovation with policy. Lin received a Ph.D. in education from Beijing Normal University. Contact him at chlin@zju.edu.cn.

7. Y. Wan, W. Zhao, H. Zhang, Y. Sui, G. Xu, and H. Jin, "What do they capture? A structural analysis of pre-trained language models for source code," in *Proc. 44th Int. Conf. Softw. Eng.*, pp. 2377–2388, 2022, doi: [10.1145/3510003.3510050](https://doi.org/10.1145/3510003.3510050).
8. Z. Kotti, R. Galanopoulou, and D. Spinellis, "Machine learning for software engineering: A tertiary study," *ACM Comput. Surv.*, vol. 55, no. 12, pp. 1–39, Nov. 2022, doi: [10.1145/3572905](https://doi.org/10.1145/3572905).
9. S. Martínez-Fernández et al., "Software engineering for AI-based systems: A survey," *ACM Trans. Software Eng. Method.*, vol. 31, no. 2, pp. 1–59, Apr. 2022, doi: [10.1145/3487043](https://doi.org/10.1145/3487043).
10. D. Russo, "Navigating the complexity of generative AI adoption in software engineering," *ACM Trans. Software Eng. Method.*, vol. 33, no. 5, pp. 1–50, Mar. 2024, doi: [10.1145/3652154](https://doi.org/10.1145/3652154).
11. R. Anwar, and M. B. Bashir, "A systematic literature review of AI-based software requirements prioritization technique," *IEEE Access*, vol. 11, pp. 143,815–143,860, 2023, doi: [10.1109/ACCESS.2023.3343252](https://doi.org/10.1109/ACCESS.2023.3343252).
12. P. Kruchten, P. Lago, and H. van Vliet, "Building up and reasoning about architectural knowledge," in *Proc. 2nd Int. Conf. Qual. Softw. Archit.*, pp. 43–58, 2006, doi: [10.1007/11921998_8](https://doi.org/10.1007/11921998_8).
13. A. Tang, P. Avgeriou, A. Jansen, R. Capilla, and M. Ali Babar, "A comparative study of architecture knowledge management tools," *J. Syst. Softw.*, vol. 83, no. 3, pp. 352–370, Mar. 2010, doi: [10.1016/j.jss.2009.08.032](https://doi.org/10.1016/j.jss.2009.08.032).
14. M. Wiese, P. Rachow, M. Riebsch, and J. Schwarze, "Preventing

- technical debt with the TAP framework for Technical Debt Aware Management," *Inf. Softw. Technol.*, vol. 148, Aug. 2022, Art. no. 106926, doi: [10.1016/j.infsof.2022.106926](https://doi.org/10.1016/j.infsof.2022.106926).
- 15. L. Argote and M. Hora, "Organizational learning and management of technology," *Prod. Oper. Manage.*, vol. 26, no. 4, pp. 579–590, Dec. 2016, doi: [10.1111/poms.12667](https://doi.org/10.1111/poms.12667).
 - 16. S. Kudaravalli, S. Faraj, and S. L. Johnson, "A configurational approach to coordinating expertise in software development teams," *MIS Quart.*, vol. 41, no. 1, pp. 43–64, Jan. 2017, doi: [10.25300/MISQ/2017/41.1.03](https://doi.org/10.25300/MISQ/2017/41.1.03).
 - 17. M. Zahedi, M. Shahin, and M. Ali Babar, "A systematic review of knowledge sharing challenges and practices in global software development," *Int. J. Inf. Manage.*, vol. 36, no. 6, pp. 995–1019, Dec. 2016, doi: [10.1016/j.ijinfomgt.2016.06.007](https://doi.org/10.1016/j.ijinfomgt.2016.06.007).
 - 18. V. Garousi, M. Felderer, and T. Hacaloglu, "What we know about software test maturity and test process improvement," *IEEE Softw.*, vol. 35, no. 1, pp. 84–92, Jan./Feb. 2018, doi: [10.1109/MS.2017.4541043](https://doi.org/10.1109/MS.2017.4541043).
 - 19. M. H. Jarrahi, D. Askay, A. Eshraghi, and P. Smith, "Artificial intelligence and knowledge management: A partnership between human and AI," *Bus. Horizons*, vol. 66, no. 1, pp. 87–99, Mar. 2022, doi: [10.1016/j.bushor.2022.03.002](https://doi.org/10.1016/j.bushor.2022.03.002).
 - 20. A. Murillo et al., "Understanding and designing for trust in AI-powered developer tooling," *IEEE Softw.*, vol. 41, no. 6, pp. 23–28, Nov./Dec. 2024, doi: [10.1109/MS.2024.3439108](https://doi.org/10.1109/MS.2024.3439108).
 - 21. S. Baltes, R. Kiefer, and S. Diehl, "Attribution required: Stack overflow code snippets in GitHub projects," in *Proc. 39th Int. Conf. Softw. Eng. Companion*, pp. 161–163, 2017, doi: [10.1109/icse-c.2017.99](https://doi.org/10.1109/icse-c.2017.99).
 - 22. W. Ma, Y. Song, M. Xue, S. Wen, and Y. Xiang, "The 'code' of ethics: A holistic audit of AI code generators," *IEEE Trans. Dependable Secure Comput.*, vol. 21, no. 5, pp. 4997–5013, Sep./Oct. 2024, doi: [10.1109/tdsc.2024.3367737](https://doi.org/10.1109/tdsc.2024.3367737).



**IEEE TRANSACTIONS ON
BIG DATA**

► SUBSCRIBE AND SUBMIT

For more information on paper submission, featured articles, calls for papers, and subscription links visit: www.computer.org/tbd



TBD is financially cosponsored by IEEE Computer Society, IEEE Communications Society, IEEE Computational Intelligence Society, IEEE Sensors Council, IEEE Consumer Electronics Society, IEEE Signal Processing Society, IEEE Systems, Man & Cybernetics Society, IEEE Systems Council, and IEEE Vehicular Technology Society

TBD is technically cosponsored by IEEE Control Systems Society, IEEE Photonics Society, IEEE Engineering in Medicine & Biology Society, IEEE Power & Energy Society, and IEEE Biometrics Council





Digital Object Identifier 10.1109/MC.2025.3573130

30 COMPUTER

WWW.COMPUTER.ORG/COMPUTER



Generative AI in Software Development: Challenges, Opportunities, and New Paradigms for Quality Assurance

F. Muratovic, Deloitte Consulting LLP

D. Kearns-Manolatos and A. Alibage , Deloitte Services LP

This research presents an overview of generative artificial intelligence's impact across the software development lifecycle. Through interviews, case study analysis, a thorough literature review, and original data analysis from proprietary Deloitte survey data, this research proposes strategic actions product, engineering, and technology leaders can take today.

Generative artificial intelligence (GenAI) has rapidly emerged as a transformative force in software development, delivering significant productivity gains, enabling process automation, and unlocking innovative design capabilities. However, its swift adoption has also revealed gaps in governance, explainability, and cost management, underscoring the need

for comprehensive strategies to harness its potential while addressing associated risks.

GenAI could revolutionize the software development lifecycle (SDLC) by enhancing productivity at every stage, from planning and design to implementation, testing, deployment, and optimization. For instance, some companies have reported productivity increases of up to 20% using autocoding tools.¹

According to Gartner, "GenAI will spawn new roles in software engineering and operations, requiring 80% of the engineering workforce to upskill."²

While GenAI tools can accelerate coding and automate repetitive tasks, their reliance on complex algorithms raises concerns about code reliability, model transparency, and process integrity.³ Furthermore, the absence of standardized metrics and frameworks for implementing GenAI within the SDLC highlights a pressing research challenge.

This research focuses on this gap by proposing actionable frameworks informed by 25 interviews, case study analysis, a thorough literature review, and original data analysis from proprietary Deloitte survey data. It explores GenAI's impact on software quality and the SDLC at each stage and explores emerging strategies for fostering human-in-the-loop (HITL) teams to enhance transparency, redefine roles, revamp testing protocols, and implement new frameworks to manage risk and cost.

METHODOLOGY

To assess GenAI's impact on the SDLC, this study utilized a mixed-methods approach involving the following methods:

- **Literature review:** We analyzed the implications of using GenAI for software development and within digital products to understand its impact on the following SDLC phases, which are recognized universally as the standard flow for modern software development: software planning and design, implementation, testing, deployment, and optimization.
- **Proprietary survey:** The proprietary survey included responses from 2,200 executives spanning industries such as technology, health care, and

manufacturing. Participants included product managers, engineers, and technology leaders, with a balanced representation across company sizes, all of which had at least US\$500 million in revenue. The survey used Likert-scale questions to measure perceptions of productivity gains, quality improvements, and adoption challenges. To maintain data reliability, responses were cross-validated with performance metrics from industry benchmarks.

- **Case studies:** We conducted in-depth analyses of specific, real-world applications of GenAI within organizations.
- **Specialist interviews:** Between July and August 2024, 25 specialists in AI, software engineering, and governance were interviewed in structured interviews to provide qualitative insights into challenges and strategies for ensuring the quality deployment of GenAI across the SDLC.

GENAI POSES NEW CHALLENGES TO SOFTWARE QUALITY ACROSS THE SDLC

GenAI is redefining software development, enhancing tasks from product road mapping to developer productivity.⁴ Deloitte's recent "State of Generative AI in the Enterprise" report shows that nearly 70% of those surveyed stated that their organizations have progressed 30% of their GenAI pilot projects into full production, with many already realizing measurable productivity gains.⁵ Additionally, a survey by BairesDev in July 2024 found that nearly 72% of software engineers

are leveraging GenAI capabilities in their development processes, with 48% using them daily. Additionally, 74% of respondents reported that GenAI has increased their quality of work to some extent, with 53% noting improvements between 10% and 25%.⁶

Defining software quality

The authors define software quality as "a product's capacity to meet requirements and user expectations, emphasizing attributes like robustness, reliability, efficiency, simplicity, security, and dependability."⁷ Quality software reliably performs its intended functions; maintains security against vulnerabilities; operates efficiently with minimal resource waste; and remains easy to manage where pertaining to its code, data, model, architecture, or functions.⁸

Leaders should consider software quality at each stage of the SDLC, including the impacts and practices of planning, implementation, testing, deployment, and optimization.

As GenAI is increasingly embedded into software and becomes part of the development process, getting ahead of software quality challenges can be a key differentiator across each of these four consolidated stages: planning and designing, implementation, testing, and deployment and optimization. This can help enhance productivity and performance and manage costs (see Table 1). For the purposes of this analysis, we consolidate software planning and design into a single stage as AI-driven tools often blur the distinction between these two phases by integrating automated documentation, architecture design, and prototyping into the early phases of development.

TABLE 1. Stages, tools, and challenges.

SDLC stage	Tools	Sample of research-identified challenges
Planning and designing	Requirement gathering; automated documentation; prototyping; architecture design	Lack of creativity; homogenized outcomes; transparency; explainability
Implementation	Autocoding tools; debugging tools	Code accuracy; technical debt
Testing	AI-generated test cases	Variability in outcomes; early integration
Deploying and optimizing	Continuous integration/ continuous delivery pipelines; analytics; optimization models	Governance; accountability; scalability; cost management

Source: Deloitte Center for Integrated Research.

PLANNING AND DESIGNING STAGE: MOVING BEYOND HOMOGENEOUS OUTCOMES THROUGH ITERATIVE DESIGN

GenAI can help product and engineering leaders summarize requirement meetings, automate and standardize documentation, and design software architectures.⁷ However, scaling GenAI may result in homogenized outputs due to reliance on existing patterns, which can lack human creativity⁹ at a critical visioning stage of software design that often supports budgeting and business buy-in.

To counteract these risks, keep design transparency and system explainability at the forefront, and consider how and where GenAI can add business value (not just cost) to a project. These two case examples emphasize this point.

IMPLEMENTATION STAGE: EXECUTING CODE ACCURACY CHECKS THROUGH HUMAN OVERSIGHT AND AUTOMATED VALIDATION

While GenAI offers productivity gains for coding,¹⁰ there is a risk of

producing overly complex, overly simplified, or inaccurate code,¹¹ which can increase technical debt and hinder system scalability.¹² Overly simplified AI-generated code, for example, may perform poorly in production environments, lacking exception handling for edge cases, capacity for peak user loads, or preparedness for unusual data formats.¹³ This can have significant financial implications as development and production costs, which will be discussed later.

Empirical research from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, reveals that code accuracy corresponds to prompt complexity. Simple prompts show a 90% code accuracy rate, whereas more complex prompts result in only 42% accuracy¹⁴ given large language models can struggle with complex, unfamiliar problems.

It is important to keep in mind that GenAI tools can demonstrate varied performance across programming languages. For instance, Python and JavaScript show high code accuracy

due to their widespread training data availability, while languages like C and SQL exhibit lower performance due to their niche use cases.^{14,15} In a recent study from Shanghai Jiao Tong University, Python achieved a 90% code accuracy rate, while SQL-based tasks showed only 72%.¹⁴ These disparities necessitate tailored approaches, including model retraining for underperforming languages and additional testing in specialized domains.¹⁶

Overall, many organizations are still learning to find the right medium. Deloitte's Q1 "State of Generative AI in the Enterprise" study, which surveyed 2,835 global business and technology leaders, found that of the 1,284 respondents using GenAI for coding, 38% expressed doubts about the results.¹⁷ Therefore, both simple and complex coding should include human oversight,¹⁴ including increased rigor related to code accuracy assessments grounded by data and implemented with a HITL model, ensuring responsibility and accountability remain with human reviewers.

Some companies have explored introducing new code review metrics, such as acceptance and rejection rates for repository submissions. According to a recent study by ComputerWorld,¹⁸ generally, on the low end, acceptance rates of GenAI coding tools that produce code vary between 30% and 40%, and on the high end, the rate is up to 50% to 80%.¹⁹

Another metric companies have turned to is changes in their application programming interface (API) leakage rates, according to the OWASP API Security Project.²⁰

API leakage can indicate security vulnerabilities, particularly in systems where access control mechanisms are improperly implemented

APPLYING AI TO SOFTWARE DEVELOPMENT

or where unintended data exposure occurs due to poor API governance. Additionally, uncommitted code volume, tracked via tokenizers, can help developers measure and monitor for negative impacts such as tech debt accumulation. High volumes of uncommitted code can often signal poor code management practices, such as excessive rewrites, unclear requirements, or unstructured experimentation without integration into the main codebase. When developers continuously generate but do not commit code, it may indicate the following:

- **Refactoring inefficiencies:** Poorly structured code requires repeated revisions before integration, increasing development time and maintenance complexity.²¹

- **Lack of clear requirements:** Developers may hesitate to commit code due to changing specifications or unclear acceptance criteria, leading to wasted effort.²²
- **Accumulation of orphaned or redundant code:** Excessive uncommitted code can result in outdated or unused functions that contribute to complexity and future debugging challenges.²³

Over time, these issues increase technical debt, slow down feature releases, and introduce risks of inconsistent system behavior, making it important to monitor uncommitted code trends alongside other software quality metrics. **Table 2** (adapted from additional sources^{24,25,26,27}) provides a sample of emerging metrics

frameworks such as defect density, time to delivery, and cost savings.

In terms of the *code review model*, HITL approaches were prevalent across almost all of the original interviews conducted for the research; however, how those teams manifest can differ by role.

Junior developers, for instance, can gain substantial benefits from GenAI but also require guidance to enable code quality and relevance. By overseeing AI-generated code, senior developers can help junior colleagues understand standards and leading practices that GenAI may overlook. As their productivity increases with GenAI support, they can tackle higher-level architecture, innovation, and engineering tasks.^{28,29}

This mentorship can help foster both skill development and quality assurance as senior developers can share insights gained through real-world experience. In this collaborative workflow, senior developers can also take on a critical role in feeding coding standards into GenAI solutions, resulting in better overall output. As productivity increases, they can tackle higher-level architecture, innovation, and engineering tasks.

Successful GenAI integration in development teams relies on a collaborative approach between junior and senior developers.

TESTING STAGE: CREATING PROMPT ENGINEERING CENTERS OF EXCELLENCE

In test engineering, the objective has traditionally been to troubleshoot many scenarios before moving software into production. GenAI has shown promise in producing a variety of AI-generated test cases. However, in a departure from the traditional SDLC workflow, our interviewees suggest

TABLE 2. Code quality metrics examples.

Metric	Definition	Measurement method	Example/application
Defect density	Number of defects per unit of code or function	Defects/kilos of lines of code (KLOC) (thousands of lines of code)	Observed reduction from 3.2 to 1.8 defects/KLOC post-GenAI use ²⁴
Time to delivery	Time taken to deliver a project/module	End-to-end duration from initiation to completion	Reduced delivery time from six weeks to four weeks using GenAI for autocoding ²⁵
Cost savings	Financial savings from reduced rework and optimized processes	Cost analysis of rework and testing	Saved US\$250,000 by minimizing rework with AI-driven testing and review ²⁵
API leakage rate	Percentage of API changes that result in unintended data leakage	Tracked through automated monitoring systems	Reduced API leakage rates by 20% through rigorous GenAI-assisted code reviews ²⁶
Uncommitted code volume	The volume of uncommitted or abandoned code in repositories	Measured via token analysis tools	Identified and eliminated redundant code, reducing repository size by 15 percentage points ²⁷

Source: Deloitte Center for Integrated Research, based on insight from additional sources.^{24,25,26,27}

testing should be done at the early product design stage instead.

GenAI solutions can yield vastly different outputs due to randomness and their mathematical design, complicating testing processes if they remain too far downstream. Testing reliability can be improved when testers work alongside engineers to fine-tune based on the model's parameters or limitations.⁵

In this way, our research finds that some companies have started to move traditional testing upstream into the software development process. They have established centers of excellence for prompt engineering, a practice of designing and refining inputs queries to optimize GenAI model outputs,³⁰ to help implement training across different user personas.

In some cases, test personas were shown to align with pretested prompts, so users only interact with prompts that have been validated in advance by creating a controlled and reliable interaction environment. For example, some companies have implemented structured workflows where domain experts design and prevalidate prompts tailored to specific user roles, such as customer support agents, developers, or analysts. These prompts can be rigorously tested for effectiveness and reliability in various scenarios before deployment, ensuring consistent and accurate interactions across diverse use cases.

A prompt library managed through a dedicated group of leaders comprising the Center of Excellence (COE) can help to narrow users' focus to a well-defined "known universe," reducing the potential for unexpected or inaccurate outputs, serving as a centralized hub for reusable prompt libraries and personas, enhancing scalability, and ensuring consistency through systematic testing.⁵ In

large-scale projects, one interviewee spoke about dividing prompts into distinct, manageable units that allow for efficient integration of independent components, with built-in safeguards to minimize inconsistencies.

While prompt engineering is still an emerging discipline, targeted training is important for anyone interacting with GenAI to understand its parameters, including business users and even customers.

Some organizations employ a hub-and-spoke model, with a core team of prompt engineers as the "hub" and business units as "spokes." As solutions scale effectively, prompts are adapted to meet specific needs. This approach aligns with leading practices in prompt engineering.⁷

Given that the COE and hub-and-spoke approaches may contradict traditional development, security, and operations (DevSecOps) processes, new governance structures should be defined to help enable human accountability, especially if GenAI inadvertently introduces flawed code.

Compressing continuous integration and delivery processes can become more critical for agile teams. One way to do this might be to bring test engineers into the initial product design phase to create and test proposed solutions simultaneously.

While these rules are still being written, those we interviewed suggest that for now, the senior engineer who currently assesses technical quality will need to extend that responsibility to GenAI-generated code, with the product owner accountable for any financial implications. In the cross-functional "pod" model, shared accountability among team members can strengthen adaptability, an essential factor for GenAI-driven projects, where the lines

across roles and functions are becoming increasingly blurred.

Most leaders interviewed agree that new governance frameworks will be needed in the future. Additionally, Deloitte's Q3 "State of Generative AI in the Enterprise" report indicates that 56% of organizations surveyed identify governance frameworks as essential for trust and risk management in GenAI applications.⁵ A structured governance approach reflects a growing trend toward managing GenAI risks proactively, enhancing trust across workflows.⁵ Training is another critical element, with 38% of companies surveyed citing trust-building through guidelines and 42% emphasizing accuracy to mitigate risks.⁷

DEPLOYING AND OPTIMIZING STAGE: GOING TO AND BEYOND PRODUCTION

As organizations deploy software with or enabled by GenAI, they should project out both short-and long-term costs across model training, deployment, and maintenance, as well as the computing infrastructure needed to run the software when scaled. Depending on whether the software is performing general-purpose tasks or using custom-trained models, there will be different demands on training and infrastructure.^{31,32}

The development costs for GenAI applications can vary widely based on complexity and features. For the basic GenAI applications, the development costs typically range from US\$20,000 to US\$150,000.³³ However, for advanced GenAI applications, for more feature-rich applications, costs can range between US\$100,000 and US\$500,000.³⁴ These estimates encompass various factors, including planning, design, AI model development, and integration. It is important

to note that actual costs can vary based on specific project requirements and the chosen development approach.

However, they may lack the flexibility and scalability required for specialized applications.³⁵ In contrast, custom-trained models demand robust infrastructure, specialized expertise, and continuous retraining, which can substantially increase expenses.

These figures underscore the variability in cost and the need for strong financial operations (FinOps) discipline. These costs include data storage, computational power, and the human resources needed for implementation and maintenance.³⁶ Conducting a thorough cost-benefit analysis is important to align investments with long-term objectives.³⁵

The transformative potential of GenAI in the SDLC is evident across each phase. Yet, as GenAI integrates more deeply into development processes, challenges arise that should not be ignored. Issues such as homogenous design, variability in code quality, and the need for new testing practices demand strategic responses and continuous adaptation.

Emerging leading practices such as HITL models, a design approach where human judgment informs decisions in automated processes, prompt engineering, and advanced code review metrics, will play a critical role. By embracing these evolving practices, the business can better leverage GenAI's potential, paving the way for a future where automation and quality coalesce seamlessly.

SUMMARY OF ACTIONS

GenAI impacts each of the stages of the SDLC. Leaders can consider several actions, including the following.

Enable design transparency and explainability

GenAI integration can drive homogenized thinking at the design stage. Business leaders should implement HITL practices and iterative design principles to confirm that products are standardized and creative, achieve funding, and align with business objectives.

Maintain code accuracy and evolve metrics as well as the developer experience

Leaders should implement strategies to help manage and reduce this debt by conducting regular codebase audits, establishing refactoring protocols, and utilizing automated testing. These practices help maintain long-term software quality and reduce maintenance burdens over time and will require junior and senior developer roles to evolve. This research suggests several metrics to distinguish effective code within GenAI-augmented systems. Organizations can adopt repeatable metrics, including the following:

- **Defect density:** number of defects per 1,000 lines of code to assess quality¹⁵
- **Time to delivery:** average time to complete development tasks¹⁶
- **Adoption metrics:** percentage of developers actively using GenAI tools.¹⁶

Revamp testing to factor prompt engineering and training

Prompt engineering is a critical skill with COEs emerging to house prompt libraries, run training, and serve as a knowledge hub for iterative development and delivery.

Leaders should define governance frameworks that address GenAI integration needs across roles and encourage cross-functional collaboration by creating agile, multidisciplinary teams with increasingly integrated roles across SDLC functions and humans and machines.

Consider continuous improvements in deployment and production

Teams should remain adaptable, imbuing a culture that values continuous learning and encourages team members to stay updated on GenAI trends, improvements, and leading practices. This commitment to learning can help organizations remain resilient and innovative as GenAI technology evolves. As organizations engage with providers and operate their infrastructure, a strong FinOps discipline can help to model and manage costs.

By adopting these actions, leaders can more effectively steer their organizations toward GenAI integration that balances productivity benefits while upholding rigorous software quality. □

ACKNOWLEDGMENT

The authors thank the many Deloitte and industry specialists interviewed for this research, including but not limited to the 25 leaders interviewed. Our research relies significantly on the insights gained from the discussions with them. The authors also acknowledge Negina Rood for supporting the secondary research for this report series, Iram Parveen for project support, and Brenna Sniderman for her insights.

REFERENCES

1. I. Bousquette, "Goldman Sachs deploys its first generative AI tool

ABOUT THE AUTHORS

F. MURATOVIC is a principal and the leader of the Software Engineering and Product Development Practice, Deloitte Consulting LLP, San Francisco, CA 94105 USA. His research interests include exploring the integration of AI and DevOps to enhance software engineering practices, particularly in improving product development and operational efficiency. This includes the impact of as-a-service model innovations on business transformation and growth strategies within the technology, media, and telecom sectors. Muratovic received an MBA from Thunderbird School of Global Management, Arizona State University. Contact him at famuratovic@deloitte.com.

D. KEARNS-MANOLATOS is a research leader and senior manager at the Center for Integrated Research, Deloitte Services LP, New York, NY 10112 USA. Her research interests include AI, cloud, and cyber. Kearns-Manolatos received a master's from Fordham University. Contact her at dkearnsmanolatos@deloitte.com.

A. ALIBAGE is a research manager/subject matter specialist at the Center for Integrated Research, Deloitte Services LP, Portland, OR 97209 USA. His research interests include organizational mindfulness/resilience, digital transformation, total quality management, organizational design, and change management. Alibage received a Ph.D. in technology management from Portland State University. Contact him at aalibage@deloitte.com.

across the firm," *Wall Street J.*, Jun. 27, 2024. [Online]. Available: <https://www.wsj.com/articles/goldman-sachs-deploys-its-first-generative-ai-tool-across-the-firm-cd94369b>

2. L. Goasdouff. "Gartner says generative AI will require 80% of engineering workforce to upskill through 2027." Gartner. Accessed: Nov. 4, 2024. [Online]. Available: <https://www.gartner.com/en/newsroom/press-releases/2024-10-03-gartner-says-generative-ai-will-require-80-percent-of-engineering-workforce-to-upskill-through-2027>

3. L. Shact, B. Kreit, G. Vert, J. Holdowsky, and N. Buckley, "Four futures of generative AI in the enterprise: Scenario planning for strategic resilience and adaptability," *Deloitte Insights*, Oct. 25, 2024. [Online]. Available: <https://www2.deloitte.com/us/en/insights/topics/digital-transformation/generative-ai-and-the-future-enterprise.html>
4. N. Balani. "Future of software development: Generative AI augmenting roles, unlocking co-innovation & transforming the industry landscape." LinkedIn. 2023. Accessed: Oct. 16, 2024. [Online]. Available: <https://www.linkedin.com/pulse/future-of-software-development-generative-ai-augmenting-roles-unlocking-co-innovation-transforming-the-industry-landscape-nishant-balani>

- future-software-development
-generative-ai-augmenting-roles
-balani
5. J. Rowan, B. Sniderman, B. Ammanath, D. Jarvis, and C. Pericos. "The state of generative AI in the enterprise." Deloitte. Accessed: Oct. 10, 2024. [Online]. Available: <https://www.deloitte.com/ce/en/services/consulting/research/state-of-generative-ai-in-enterprise.html>
 6. M. Vizard. "Survey surfaces wide generative AI adoption among software engineers." DevOps.com. Accessed: Feb. 24, 2025. [Online]. Available: <https://devops.com/survey-surfaces-wide-generative-ai-adoption-among-software-engineers/#>
 7. F. Muratovic, J. Gill, D. Kearns-Manolatos, and A. Alibage, "How can organizations engineer quality software in the age of generative AI?" *Deloitte Insights*, Oct. 28, 2024. [Online]. Available: <https://www2.deloitte.com/us/en/insights/industry/technology/how-can-organizations-develop-quality-software-in-age-of-gen-ai.html>
 8. Y. Tesfay, "Developing structured procedural and methodological engineering designs," in *Quality in the Context of Engineering*, Cham, Switzerland: Springer-Verlag, 2021, pp. 3–57.
 9. A. R. Doshi and O. P. Hauser, "Generative AI enhances individual creativity but reduces the collective diversity of novel content," *Sci. Adv.*, vol. 10, no. 28, 2024, doi: [10.1126/sciadv.adn5290](https://doi.org/10.1126/sciadv.adn5290).
 10. T. Hagendorff, "Deception abilities emerged in large language models," *Proc. Nat. Acad. Sci.*, vol. 121, no. 24, 2024, Art. no. e2317967121.
 11. L. Proven, "Drowning in code: The ever-growing problem of ever-growing codebases," *The Register*, Feb. 12,

APPLYING AI TO SOFTWARE DEVELOPMENT

2024. [Online]. Available: https://forums.theregister.com/forum/all/2024/02/12/drowning_in_code/
12. T. H. Davenport and N. Mittal, "How generative AI is changing creative work," *Harvard Bus. Rev.*, 2022. [Online]. Available: <https://hbr.org/2022/11/how-generative-ai-is-changing-creative-work>
13. X. Ren, X. Ye, D. Zhao, Z. Xing, and X. Yang, "From misuse to mastery: Enhancing code generation with knowledge-driven AI chaining," in *Proc. 38th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, 2024, pp. 976–987, doi: [10.1109/ASE56229.2023.00143](https://doi.org/10.1109/ASE56229.2023.00143).
14. A. Yang, Z. Li, and J. Li, "Advancing GenAI assisted programming—A comparative study on prompt efficiency and code quality between GPT-4 and GLM-4," 2024, *arXiv:2402.12782*.
15. Z. Liu, Y. Tang, X. Luo, Y. Zhou, and L. F. Zhang, "No need to lift a finger anymore? Assessing the quality of code generation by ChatGPT," *IEEE Trans. Softw. Eng.*, vol. 50, no. 6, 1548–1584, Jun. 2024, doi: [10.1109/TSE.2024.3392499](https://doi.org/10.1109/TSE.2024.3392499).
16. K. Panetta, "Set up now for AI to augment software development." Gartner. Accessed: Oct. 19, 2024. [Online]. Available: <https://www.gartner.com/en/articles/set-up-now-for-ai-to-augment-software-development>
17. J. Rowan, B. Sniderman, B. Ammanath, D. Jarvis and C. Pericos, "The state of generative AI in the enterprise," Deloitte, London, England, 2024. [Online]. Available: <https://www2.deloitte.com/us/en/pages/consulting/articles/state-of-generative-ai-in-enterprise.html>
18. L. Mearian, "Just how good is AI-assisted code generation?" *ComputerWorld*, Apr. 3, 2024. [Online]. Available: <https://www.computerworld.com/article/2077802/just-how-good-is-ai-assisted-code-generation.html>
19. F. Muratovic, D. Stewart and P. Raman, "Tech companies lead the way on generative AI: Does code deserve the credit?" *Deloitte Insights*, Aug. 2, 2024. [Online]. Available: <https://www2.deloitte.com/us/en/insights/industry/technology/gen-ai-coding-tools.html>
20. "OWASP API security project." OWASP.org. Accessed: Oct. 19, 2024. [Online]. Available: <https://owasp.org/www-project-api-security/>
21. A. Sursaieva, "Best practices for managing technical debt effectively," *Axon Develop.*, Jun. 12, 2024. [Online]. Available: <https://www.axon.dev/blog/best-practices-for-managing-technical-debt-effectively>
22. P. Gillin, "What is technical debt? Examples, prevention & best practices," *Mendix*, Aug. 12, 2024. [Online]. Available: <https://www.mendix.com/blog/what-is-technical-debt/>
23. "From Chaos to Clarity: Effective technical debt management tactics." DMI. Accessed: Nov. 15, 2024. [Online]. Available: <https://dminc.com/insight/from-chaos-to-clarity-effective-technical-debt-management-tactics/>
24. "KPIs for product managers: Defect density." Ignition GTM, Inc. Accessed: Feb. 12, 2025. [Online]. Available: <https://www.haveignition.com/kpis-for-product-managers/kpis-for-product-managers-defect-density>
25. Flow Transformation Team, "13 top software development KPIs you should track in 2024," Pluralsight, Jan. 22, 2025. [Online]. Available: <https://www.pluralsight.com/resources/blog/software-development/software-development-KPIs>
26. T. Cser, "10 key KPIs driving continuous improvement in QA teams," *Functionize*, Dec. 6, 2023. [Online]. Available: <https://www.functionize.com/blog/10-key-kpis-driving-continuous-improvement-in-qa-teams>
27. Engineering Team, "25 software development KPIs with examples," *ClickUp*, Jan. 27, 2025. [Online]. Available: <https://clickup.com/blog/software-development-kpis/>
28. J. Gill and R. Ravi, "Horizon architecture: Business strategy activation: The hidden superpower for activating your enterprise business strategy." Deloitte. Accessed: Nov. 2, 2024. [Online]. Available: <https://www2.deloitte.com/us/en/pages/strategy-operations/articles/enterprise-business-strategy-architecture.html>
29. S. Aulbach, B. Langmack, M. Lachmann, M. Eberle and Y. Zeng, "AI-powered application rewrite: Revolutionizing legacy code transformation." Deloitte. Accessed: Nov. 2, 2024. [Online]. Available: <https://www2.deloitte.com/us/en/pages/consulting/articles/ai-powered-application-rewrite-revolutionizing-legacy-code-transformation.html>
30. N. Runyon, "Getting the best of GenAI: How to use prompt engineering," Thomson Reuters, Dec. 4, 2024. [Online]. Available: <https://www.thomsonreuters.com/en-us/posts/ai-in-courts/prompt-engineering/>
31. T. H. Davenport and N. Mittal, "How generative AI is changing creative work," *Harvard Bus. Rev.*, Nov. 12, 2022. [Online]. Available: <https://hbr.org/2022/11/how-generative-ai-is-changing-creative-work>

32. J. Moore. "CIOs weigh generative AI costs as ROI comes into focus." TechTarget. Accessed: Oct. 9, 2024 [Online]. Available: <https://www.techtarget.com/searchcio/feature/CIOs-weigh-generative-AI-costs-as-ROI-comes-into-focus>
33. K. Wadhwan. "How much does it cost to build a generative AI in 2025." SoluLab. Accessed: Feb. 15, 2025. [Online]. Available: <https://www.solulab.com/cost-of-building-a-generative-ai-solution/#:~:text=Creating%20generative%20>
34. S. Dhruv, "How much does it cost to build a generative AI?" Aalpha, Dec. 26, 2024. [Online]. Available: <https://www.aalpha.net/blog/cost-to-build-a-generative-ai/#:~:text=The%20development%20of%20Generative%20AI,range%20between%20%24600%2C000%20to%20%241%2C500>
35. "GenAI and ROI: Building the business case." BDO USA. Accessed: Feb. 8, 2025. [Online]. Available: <https://www.bdo.com/insights/advisory/genai-and-roi-building-the-business-case>
36. V. Gupta, "Quantifying the ROI of generative AI with a focus on cost-efficiency," Forbes, Apr. 14, 2024. [Online]. Available: <https://www.forbes.com/councils/forbestechcouncil/2024/08/14/quantifying-the-roi-of-generative-ai-with-a-focus-on-cost-efficiency/>

Call for Articles

IEEE Pervasive Computing

seeks accessible, useful papers on the latest peer-reviewed developments in pervasive, mobile, and ubiquitous computing. Topics include hardware technology, software infrastructure, real-world sensing and interaction, human-computer interaction, and systems considerations, including deployment, scalability, security, and privacy.

Author guidelines:
www.computer.org/mcpervasive/author.htm

Further details:
pervasive@computer.org
www.computer.org/pervasive

IEEE **pervasive**
COMPUTING
MOBILE AND UBIQUITOUS SYSTEMS



AI for Productivity: Transforming Enterprise Software Development

Amit Jain  and **Rohit Tiwari** , DevOn Software Services

Through a survey study and real-world examples, this study explores how artificial intelligence (AI)-powered project management, code generation, automated testing, and AI-enhanced DevOps are revolutionizing development workflows. The article also highlights best practices for implementing AI effectively and the key considerations and challenges involved, along with an analysis of the emerging trends and the potential of AI in driving continuous improvement in enterprise environments.

Artificial Intelligence (AI) productivity in software development refers to the application of AI techniques and tools to augment, streamline, and automate various phases of

the software development lifecycle (SDLC). The SDLC process generally encompasses several essential phases, each playing a crucial role in the successful creation of software. The primary phases of the SDLC include planning, implementation, testing, and deployment, though this is not an exhaustive list.¹ By integrating AI into development workflows, teams can optimize processes,

Digital Object Identifier 10.1109/MC.2025.3543181
Date of current version: 27 June 2025

reduce manual effort, and concentrate on high-impact, strategic tasks. This integration facilitates efficiency, accuracy, and innovation across software projects.

AI productivity supports development teams in multiple ways. By automating repetitive tasks such as code generation, testing, and debugging, it allows developers to focus on complex, high-priority challenges.² It enhances code quality and maintainability through the detection of code smells, enforcement of best practices, and support for refactoring.³ AI-driven optimization techniques improve software performance by adapting to real-time data and usage patterns. Furthermore, machine learning (ML) models trained on historical bug and security data aid in predicting and mitigating software defects and vulnerabilities. AI also contributes to better resource allocation and project management by providing actionable insights to help teams plan, prioritize, and manage tasks effectively.

In summary, AI productivity embeds intelligent automation and analytics into the development process, fostering the creation of faster, more reliable, and higher-quality software.

STRUCTURE OF THE ARTICLE

This article examines the impact of AI tools on productivity across various stages of the SDLC. The discussion begins with an exploration of the benefits of AI tools and technologies in different SDLC phases, supported by contemporary data on existing tools. Next, best practices and strategies for implementing AI-driven software development are presented.

The subsequent section analyzes the results of a survey study assessing the productivity impact of AI tools in

different SDLC stages. This is followed by a discussion on key considerations for AI adoption in enterprise software development. Finally, the challenges and emerging trends in AI-driven productivity are examined, followed by a conclusion.

AI TOOLS AND TECHNOLOGIES FOR ENTERPRISE SOFTWARE DEVELOPMENT

The rapid evolution of AI tools and technologies is reshaping enterprise software development, enabling teams to work more efficiently, improve code quality, and accelerate project timelines. From AI-driven code completion to predictive project management and intelligent testing, these tools cover a wide range of functions that enhance productivity at every stage of the SDLC.³

AI-powered project management tools

AI-powered project management tools utilize advanced ML algorithms to enhance resource allocation, task management, and project planning. They also provide intelligent recommendations for optimizing task assignments, scheduling, and resource distribution. They not only automate routine project management tasks but also predict potential bottlenecks, identify risks, and suggest adjustments to keep projects on track. By leveraging AI for decision-making, organizations can improve operational efficiency, foster better collaboration, and ensure more accurate, timely project deliveries. A practical example is Atlassian Intelligence, which reports that weekly users save over 45 min on average, with 77% of users experiencing time savings due to AI-powered

search functions.⁴ Some of the popularly used AI-powered project management tools are ClickUp,⁵ Jira,⁶ Asana,⁷ Trello,⁸ Notion,⁹ and so on.

AI-powered code generation and autocompletion tools

AI-powered code generation and autocompletion tools significantly enhance developer productivity by automating repetitive coding tasks and providing contextually relevant suggestions. Leveraging advanced ML algorithms, these tools analyze existing code patterns, libraries, and application programming interface (API) documentation to generate code snippets, autocomplete lines or entire code blocks, and suggest optimized coding solutions. By reducing the time spent on manual coding and minimizing the likelihood of syntax errors, these tools allow developers to focus on higher-level tasks such as architecture design and problem-solving. For instance, CI&T saw an impressive 11% increase in productivity by using Tabnine, with developers accepting 90% of the tool's single-line coding suggestions.¹⁰ Similarly, Copilot integration led to a 10.6% increase in pull requests and reduced cycle times by 3.5 h,¹¹ underscoring the efficiency gains that AI-driven code completion can bring to the development lifecycle. As the AI learns from vast codebases and developer preferences, it becomes increasingly adept at making accurate suggestions tailored to specific projects and coding styles, further accelerating the development process. Some of the notable AI-driven code generation and autocompletion tools are GitHub Copilot,¹² Tabnine,¹³ Amazon Q,¹⁴ Gemini Code Assist,¹⁵ Cursor,¹⁶ and so on.

AI-driven code review has transformed software quality assurance by leveraging ML to detect bugs, code smells, and security vulnerabilities automatically. Studies have shown that these tools enhance developer productivity by providing early and actionable feedback.^{17,18} A prominent example is CodeRabbit, which employs transformer-based models to identify issues such as cyclic dependencies, unused code, and inefficient loops. CodeRabbit is reported to deliver 86% faster code delivery, reducing turnaround time from days to hours, and decreases code review issues by over 60%.^{19,20} Additionally, it achieves a 70% implementation rate of AI-powered fixes and processes millions of pull requests monthly, ensuring unified quality assurance and automated security checks.²⁰ By combining contextual code understanding with automation, tools like CodeRabbit are becoming indispensable in modern development workflows.

AI-powered automated testing and bug detection tools

AI-powered tools are revolutionizing the testing phase by automating test case generation, enhancing bug detection, and improving overall test coverage. These tools generate test cases that prioritize critical code paths, ensuring more comprehensive test coverage and adaptability, as the tools continuously evolve based on previous tests and developer feedback. By detecting bugs earlier in the development cycle, these tools not only reduce costly post-deployment issues but also enhance code quality and accelerate the delivery of reliable software. A notable example is Medrio's experience with testRigor, where AI-enabled testing scaled their release frequency from two to eight

times per year, a fourfold increase. Additionally, Medrio saved approximately four man-years annually in manual quality assurance (QA) efforts, highlighting the efficiency gains achievable with AI-driven testing.²¹ Some of the notable AI-driven testing and bug-detection tools are Testim,²² Applitools,²³ Katalon,²⁴ LAMBDA-EST,²⁵ ACCELQ,²⁶ and so on.

AI-powered DevOps and continuous integration/continuous delivery tools

AI technologies have seamlessly integrated into DevOps practices, fundamentally transforming the way software is developed, tested, deployed, and operated. AI-enhanced DevOps tools leverage ML algorithms to analyze code, monitor application performance, optimize operational efficiency, and automate critical processes throughout the software delivery pipeline. By continuously assessing system performance and user interactions, these tools can predict and preemptively address potential issues, optimize resource allocation, and enable smoother, more efficient deployments. By automating repetitive tasks such as code integration, testing, and deployment, these tools reduce manual errors and accelerate time to market. The integration of AI into continuous integration/continuous delivery (CI/CD) pipelines enhances agility, supports continuous improvement, and empowers organizations to deliver high-quality software to customers with speed and precision. A prime example is Mysa's success with Harness, where AI-driven automation improved engineering productivity by 40% and saved SRE engineers' substantial time by eliminating the need to

build custom approval workflows and manage secrets manually. Additionally, implementing "shift-left" testing for security saved 15% of engineering bandwidth, demonstrating how AI-driven DevOps streamlines complex workflows.²⁷

MLOps, a fusion of ML and DevOps, streamlines the ML lifecycle by automating repetitive tasks, ensuring robust CI/CD pipelines, and enhancing model monitoring and retraining capabilities. Key technologies include containerization with Docker, orchestration via Kubernetes, and versioning tools like DVC or MLflow for managing datasets, models, and experiments. These practices significantly reduce time-to-market for ML models while maintaining quality and scalability.^{28,29} Future advancements could include adaptive pipelines that use meta-learning to automatically adjust to changing data distributions, and enhanced model interpretability frameworks like SHAP or LIME to improve decision-making transparency.³⁰ Moreover, the integration of traceability systems³¹ ensures rigorous tracking of artifacts, enabling seamless rollback, debugging, and compliance, thus laying the groundwork for more resilient and accountable ML systems.

Some AI-enabled DevOps tools are Datadog,³² new relic,³³ Harness,³⁴ digital.ai,³⁵ Dynatrace,³⁶ and so on.

BEST PRACTICES AND STRATEGIES FOR AI-DRIVEN ENTERPRISE SOFTWARE DEVELOPMENT

Incorporating AI-driven productivity into enterprise software development requires meticulous planning, strategic implementation, and adherence to well-defined best practices. To ensure

successful adoption and maximize the benefits of AI technologies, organizations should consider the following strategies:

Identifying the right use cases for AI productivity

The first step in adopting AI in enterprise software development is to identify use cases where AI can provide significant value. A thorough analysis of business processes, pain points, and opportunities for automation is essential. Key factors to consider include the following:

Complexity and repetitiveness. Tasks or processes that are complex, time-consuming, and involve repetitive activities are ideal candidates for automation using AI technologies. For example, automated code reviews and test case generation can significantly reduce development effort and time.³⁷ While some repetitive and time-consuming tasks might generate less direct value, they often consume significant time and lead to repetitive costs for the organization. Additionally, these tasks contribute to context switching for development teams, which hampers overall productivity. Automating such tasks using AI technologies helps reduce these inefficiencies, allowing teams to focus on higher-value activities.

Data intensity. AI thrives on data. Processes reliant on large volumes of data, such as log analysis and anomaly detection, can benefit greatly from AI-driven insights.³⁸

Quantifiable metrics. Use cases should have measurable outcomes, such as reduced manual effort, improved efficiency, or enhanced code quality. For

example, implementing AI-powered defect prediction models can help quantify reductions in post-deployment issues.³⁹

By focusing on these factors, organizations can effectively channel their AI efforts and achieve tangible results.

Establishing a robust data foundation

To ensure successful AI adoption, enterprises should do the following:

- Build centralized and scalable data repositories.
- Ensure data quality, consistency, and governance.
- Protect sensitive data with robust security measures.⁴⁰

Integrating AI with Agile and DevOps practices

Integrating AI into Agile and DevOps pipelines ensures seamless collaboration and efficiency. Some of the practices include:

- incorporating AI tools for continuous integration and testing
- using AI-driven insights to prioritize tasks and allocate resources effectively
- implementing AI-based monitoring tools for real-time feedback.³⁷

Emphasizing continuous learning and upskilling

AI technologies evolve rapidly, requiring teams to stay updated. In this respect, organizations should do the following:

- Invest in training programs to enhance AI literacy.
- Encourage cross-functional collaboration to share AI knowledge.

- Establish AI Centers of Excellence to drive innovation and best practices.⁴¹

SURVEY STUDY

As part of this research, a comprehensive survey was conducted with over 250 respondents affiliated with DevOn Software Services. The respondents included product owners, scrum masters, developers, quality assurance professionals, and DevOps, representing a diverse group of professionals involved in all phases of the SDLC. Each participant is an experienced professional in enterprise software development, particularly within the U.K. and EU markets. Of the respondents, 37% have 5–10 years of experience, 32% have more than 10 years, 22% have less than one year, and 9% have 1–5 years. The structured questionnaire designed for this study assessed respondents' familiarity with and utilization of AI tools and technologies across different stages of the SDLC and also gauging both the current and potential impacts of these AI-driven tools on productivity and operational efficiency in enterprise settings. To quantify productivity effects, respondents rated the impact of AI tools on a graduated scale from 0 to 100% as follows:

- very low: less than 10%
- low: 10% to 25%
- moderate: 25% to 50%
- high: 50% to 75%
- very high: greater than 75%.

Summary of key findings

Figure 1 presents the key findings of the survey study.

Notes from the table in Figure 1 are as follows:

- For the analysis of each SDLC phase, only responses from

APPLYING AI TO SOFTWARE DEVELOPMENT

the subgroup of professionals directly involved in that specific phase were considered.

► Adoption rate refers to the familiarity in respect of the AI tools, calculated as a percentage of total respondents using one or more of the AI tools.

► The percentages in columns 4 and 5 refer to the ranges of impact on productivity reported by 75% or more of the respondents.

The subsequent sections present the crucial insights derived from the survey,

organized and categorized according to the different phases of SDLC.

AI-powered project management tools

The survey reveals that AI-powered project management tools are still in the early adoption phase, with only 14.5% of respondents currently using them, laying the groundwork for future growth. A clear preference for established platforms like Jira highlights a trend toward gradually integrating AI into existing workflows rather than replacing them. More than 60% of the respondent experienced productivity gain in the range of 25% to 50%, emphasizing AI's transformative role in project management. Additionally, 60% of respondents anticipate moderate productivity gains from AI tools, reflecting growing confidence in their long-term impact and the increasing role AI is expected to play as adoption expands and tools evolve.

The present and potential impact on productivity is graphically depicted in Figure 2.

	Adoption Percentage	Tool(s) Preference	Present Impact on Productivity	Potential Impact on Productivity
AI-Powered Project Management Tools	14.5%	Jira	10–75%	10–75%
AI-Powered Code Generation and Auto-Completion Tools	32.7%	ChatGPT, Copilot, Codex, BardAI, Tabnine, GPT-J, AWS Code Whisperer	25–100%	25–75%
AI-Powered Automated Testing and Bug Detection Tools	11.4%	Applitools	10–50%	25–100%
AI-Powered DevOps and CI/CD Tools	11.4%	Datadog, PagerDuty, Harness, Atlassian Intelligence	50–100%	25–100%

FIGURE 1. Key findings of the survey study.

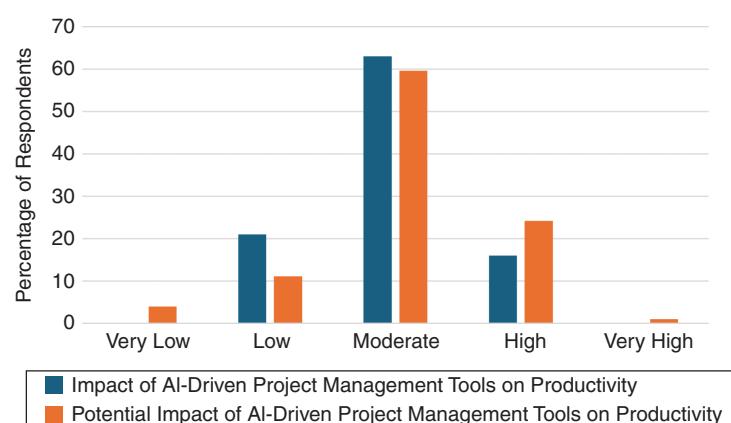


FIGURE 2. Actual versus potential productivity boost from AI-powered project management tools.

AI-powered code generation and autocompletion tools

With 32.7% of developers already integrating tools like ChatGPT, Copilot, Codex, BardAI, Tabnine, GPT-J, and AWS Code Whisperer into their workflows, code generation and auto completion shows the highest adoption rate among all SDLC phases, underscoring their role in enhancing productivity. Reported productivity gains ranging from 10% to 100% reflect the varying levels of tool utilization, from simple autocompletion to advanced code generation. Notably, 47% of respondents anticipate moderate productivity growth, while 35%

foresee significant increases, indicating strong confidence in the transformative potential of AI tools for development. The high level of current AI adoption indicates that the respondents have successfully integrated AI tools into their workflows and are reaping benefits today. Thus, the resulting gap between current and anticipated productivity gains suggests a changed baseline, where initial AI-driven boosts become the new norm, leading to more moderate future expectations.

The present and potential impact on productivity is graphically depicted in [Figure 3](#).

AI-powered automated testing and bug detection tools

The survey reveals that, while AI-driven testing tools are still in the early stages of adoption, with 11.4% of QA practitioners using them, their growing presence signals promising potential. Tools like AppliTools stand out as leaders, showcasing their effectiveness and setting a benchmark for others to follow. Notably, 50% of respondents reported productivity improvements between 25% and 50%, highlighting the efficiency and speed these tools bring to bug detection and test generation. Looking ahead, there is strong optimism about the future impact of AI-powered testing, with expectations for continued and substantial productivity gains.

The present and potential impact on productivity is graphically depicted in [Figure 4](#).

AI-powered DevOps and CI/CD tools

The survey highlights the early yet growing adoption of AI-powered tools in DevOps, with 11.4% of respondents already leveraging these solutions to

automate and optimize workflows. Tools like Datadog, PagerDuty, Harness, and Atlassian Intelligence showcase a diverse ecosystem, offering capabilities ranging from monitoring and incident response to intelligent deployment and automation. Respondents unanimously reported significant productivity gains, with many noting a remarkable 50% to 100% increase in efficiency. Looking ahead, participants expressed confidence in

the scalability and impact of AI-driven DevOps solutions, anticipating further improvements in agility, efficiency, and continuous innovation as these tools evolve.

The present and potential impact on productivity is graphically depicted in [Figure 5](#).

Limitations of the study

While the survey provides valuable insights into the productivity gains

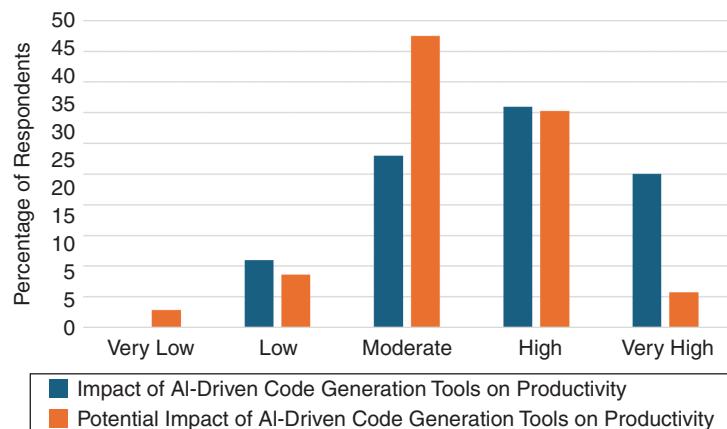


FIGURE 3. Actual versus potential productivity boost from AI-powered code generation and autocompletion tools.

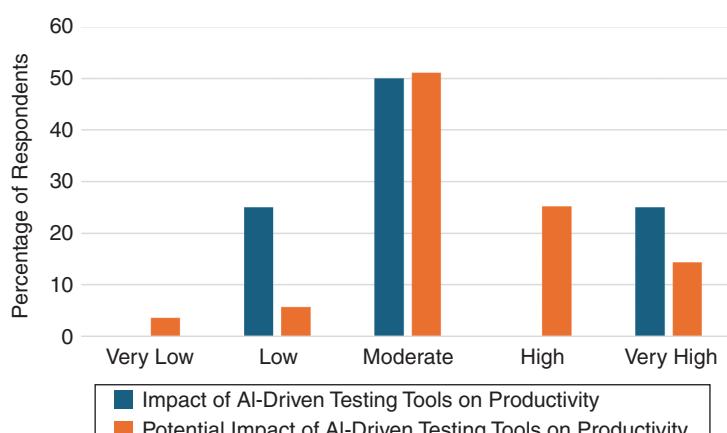


FIGURE 4. Actual versus potential productivity boost from AI-powered automated testing and bug detection tools.

APPLYING AI TO SOFTWARE DEVELOPMENT

associated with integrating AI tools and technologies across various phases of the SDLC, it has certain limitations. The study involved a relatively smaller number of respondents from a single organization, limiting the generalizability of the findings to broader populations. Additionally, the survey primarily captured participants' perceptions and experiences, without employing precise quantitative measures to assess the direct impact of AI tools on productivity. These limitations highlight the need for future research with larger and more diverse samples, combining qualitative and quantitative approaches to achieve a more comprehensive understanding of the impact.

KEY CONSIDERATIONS FOR AI IMPLEMENTATION IN ENTERPRISE SOFTWARE DEVELOPMENT

When implementing AI in enterprise software development, several critical factors must be evaluated to ensure successful adoption and meaningful outcomes. The primary considerations

are cost, value and risk, each of which requires a thorough understanding to balance opportunities and challenges effectively.

Cost

The financial investment in AI spans beyond development, encompassing infrastructure, data management, tools, and continuous training efforts. Cost includes the expense of selecting and fine-tuning large language models (LLMs), cloud infrastructure for computing, data mobilization and governance, and tools requiring engineering licenses. Additionally, training costs for end users and implementation experts, along with iterative efforts like token prompting and refinement, can significantly influence budgets. Organizations must adopt a holistic approach to cost estimation, ensuring alignment with long-term objectives and avoiding unforeseen financial burdens.⁴²

Value

Organizations must carefully measure the value derived from AI initiatives.

Efforts driven purely by fear of missing out or market hype can yield limited returns. Conversely, investments targeting moderate-value outcomes, such as enhancing development loops with code completion or expediting quality automation, can yield significant benefits. Organizations should focus on use cases that align with strategic goals, balancing innovation with tangible return on investment to maximize AI's potential.⁴²

Risk

This is an equally important dimension, with ethical and security challenges being at the forefront. Potential concerns include breaches of privacy, intellectual property leaks, productivity impacts, and damage to reputation. Effective governance frameworks, trust-building mechanisms, and transparent operations are essential to mitigate these risks. Addressing these elements is critical for establishing confidence among stakeholders while navigating moral and ethical responsibilities in AI implementation.⁴²

Having explored the critical factors of cost, value, and risk as major considerations for implementing AI in enterprise software development, it is equally important to address other vital aspects. Among these, ethical, security, and privacy considerations stand out as fundamental to ensuring responsible and effective AI adoption.

Ethical, security, and privacy considerations

Developers must consider ethical considerations as AI technologies increasingly integrate into enterprise software development. Organizations should be mindful of potential biases, privacy concerns, and the impact of

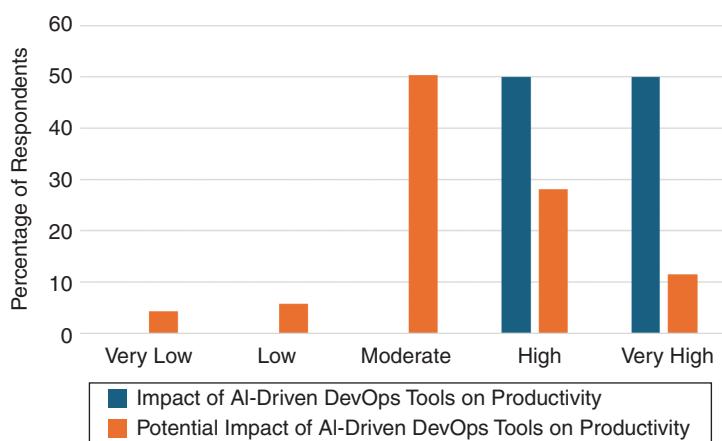


FIGURE 5. Actual versus potential productivity boost from AI-powered DevOps and CI/CD tools.

AI-driven decisions. To address ethical considerations, the following best practices can be considered.

Data privacy and security. Organizations should ensure proper data privacy and security measures are in place to protect sensitive user information and comply with relevant regulations. Safeguard sensitive data for training AI models by employing encryption, anonymization, and access controls. Comply with data protection regulations and ensure user consent for data usage.^{43,44,45,46}

Transparency and explainability. Develop AI models and algorithms that are transparent and explainable, allowing stakeholders to understand the decision-making process behind AI-driven systems.^{43,44}

Fairness and bias mitigation. Regularly assess AI models for biases⁴⁷ and take steps to mitigate them. Address fairness, diversity, and inclusion issues to ensure AI systems do not perpetuate discriminatory practices.^{43,44} Organizations should conduct security audits and vulnerability assessments of AI-enabled software systems to identify and address potential security risks.

Human oversight and accountability. Maintain human oversight and accountability⁴⁷ throughout the AI development and deployment. Establish clear guidelines for handling AI errors, biases, and unintended consequences.^{43,44,46}

Organizations can build responsible and trustworthy AI-driven software systems by considering ethical implications and incorporating safeguards.

CHALLENGES AND LIMITATIONS OF AI-DRIVEN SOFTWARE DEVELOPMENT

While AI productivity holds immense potential, it is essential to acknowledge the challenges and limitations associated with its adoption. Some of these challenges include the following.

Data quality and bias

AI models depend heavily on high-quality, representative data for accurate predictions and decision-making. For instance, an AI-powered bug prediction tool trained on biased data may overlook certain types of bugs or produce false positives in codebases that differ from the training data. If the dataset primarily includes projects in one programming language or style, the tool may perform poorly when applied to diverse environments.^{48,49}

Ethical considerations

Ethical concerns arise in areas like data privacy, security, and job displacement. For example, an AI-driven code generation tool that analyzes historical data from projects could unintentionally expose sensitive information if privacy guidelines aren't strictly enforced. Similarly, automation in tasks traditionally done by human developers might lead to concerns about workforce reduction.⁴⁹ Addressing these issues requires clear ethical frameworks, such as privacy-preserving techniques (for example, anonymization), and a commitment to reskilling programs that support affected employees.^{48,43,44}

Lack of domain expertise

Implementing AI tools effectively often demands specialized knowledge

of both software development and AI. For example, using an advanced AI model for predictive maintenance may require data scientists and engineers with expertise in ML, software engineering, and statistical analysis. Organizations without these skill sets may find it difficult to interpret AI outputs correctly or make the most of AI's potential.^{48,49}

Adoption and integration

Integrating AI tools into existing workflows often encounters resistance and technical hurdles. For instance, a team adopting an AI-based automated testing tool may need to adjust their testing framework or CI/CD pipeline, which can be costly and time consuming. Without sufficient change management, teams may resist adopting these tools due to fears of disruption or incompatibility with established processes.⁴⁹

Continual learning and adaptation

AI models need ongoing updates and adjustments to stay relevant and effective. For example, an AI tool designed to identify security vulnerabilities must be updated regularly to recognize newly discovered threats. If the tool relies on outdated patterns, it may fail to identify emerging vulnerabilities, exposing the organization to security risks.^{48,49}

Quality control of generated content

AI-generated content, such as code, documentation, or recommendations, requires rigorous quality checks. For instance, an AI-powered code suggestion tool might produce code that compiles but doesn't align with project-specific conventions or maintainability

standards. Poor quality control of AI outputs could introduce inefficiencies or even critical errors in the development cycle.^{48,49}

Intellectual property and copyright

AI-generated content raises potential concerns regarding intellectual property and copyright. Organizations must ensure at every stage of the process that the generated content adheres to legal and licensing requirements.⁴⁹

FUTURE TRENDS AND IMPLICATIONS OF AI PRODUCTIVITY IN ENTERPRISE SOFTWARE DEVELOPMENT

The field of AI productivity in enterprise software development is continuously evolving, with new technologies and trends shaping the way software is built and delivered. Some of these trends are discussed next.

Multi-agent systems

Multi-agent systems are transforming software development by enabling collaborative, role-specific intelligent agents that streamline workflows and enhance efficiency. Current implementations, such as ChatDev⁵⁰ and MetaGPT,⁵¹ simulate the operations of software companies, automating tasks from requirement gathering to deployment. Looking ahead, frameworks like Cross-Team Collaboration⁵⁰ aim to leverage LLMs to further support scalable and Agile development, paving the way for more adaptive and efficient methodologies in the future.

Enhanced project management

AI tools are increasingly pivotal in optimizing resource allocation,

accurately monitoring epic plan deviations, and delivering adaptive recommendations to align resources with project objectives. Currently, these tools assist in ensuring project success through data-driven insights and predictive analytics. In the future, advancements in AI are expected to further enhance their ability to dynamically adapt to evolving project conditions and provide even more precise strategies for achieving project goals like multicriteria decision analysis approach.⁵²

Predictive software development

Developers can apply ML techniques to predict various aspects of software development, leading to more efficient and informed decision-making. Some areas where ML can have a significant impact are as follows.

Predictive code completion. Machine learning algorithms can analyze the context of the code being written and predict subsequent lines, offering intelligent code completion suggestions. These predictive models significantly enhance developer productivity by reducing the time spent on writing boilerplate code and repetitive tasks. AI-powered code completion tools, such as GitHub Copilot,¹² have been shown to improve coding efficiency, allowing developers to focus on more complex problems rather than repetitive coding tasks.^{53,54} The future of predictive code completion will involve tools that can understand higher-level intentions and generate entire code blocks or algorithms, adapting to the individual developer's style and the unique context of a project. These advancements will extend beyond syntax to include suggestions for architectural patterns, error handling, and

logic optimization, creating a more intuitive development experience.

Predictive bug detection. Machine learning models can analyze code repositories, bug reports, and historical data to predict potential bugs or vulnerabilities. By identifying patterns and common pitfalls, these models enable developers to proactively address issues, thereby minimizing software defects. Studies have demonstrated that AI-driven bug detection tools can effectively identify issues early in the development cycle, improving code quality and reducing the cost of fixing defects.^{2,55} The way forward involves expanding these models to predict more complex and less obvious bugs, including those in integration and system-level code.

Predictive software performance.

Machine learning models can analyze code change history, bug reports, and performance metrics to predict future software performance. By continuously monitoring software performance and identifying potential areas of concern, organizations can take proactive steps to address issues before they impact users, ensuring optimal performance and user satisfaction. These predictive capabilities are essential in maintaining the long-term health and scalability of enterprise software systems.^{53,56} The future of predictive software performance will involve real-time performance tracking across entire software ecosystems, integrating performance data from various layers, including front end, back end, and infrastructure. Predictive models will evolve to suggest optimizations and capacity adjustments proactively, ensuring

that systems perform efficiently at scale.

The future of DevOps with AI

The integration of AI into DevOps is shaping the future of software development and operations by introducing intelligent automation, predictive analytics, and adaptive systems. AI-driven DevOps is poised to revolutionize traditional practices by leveraging ML models for automated software deployment, continuous monitoring, and maintenance.^{57,58} One emerging trend is the use of AI to enhance software artifact traceability, allowing teams to efficiently track changes across the development lifecycle, ensuring compliance, and reducing technical debt. Predictive analytics is also being utilized to foresee potential failures and optimize resource allocation, enabling organizations to preemptively address bottlenecks in CI/CD pipelines.^{57,58}

Moreover, adaptive feedback loops powered by AI enable dynamic adjustments to system configurations based on real-time performance metrics, ensuring system stability and scalability.⁵⁹ Future advancements in AI-driven DevOps are expected to emphasize DataOps integration, creating synergies between data management and DevOps to modernize cloud enterprise architectures and foster end-to-end automation.^{58,60} As AI continues to mature, the focus will shift toward building explainable AI models to improve trust and accountability in automated decision-making within DevOps processes. Ultimately, AI in DevOps will empower organizations to achieve unprecedented levels of agility, reliability, and efficiency in software development and operations.

ABOUT THE AUTHORS

AMIT JAIN is the managing director at DevOn Software Services, Delft, 2612 CT, The Netherlands. His research interests include artificial intelligence (AI) in software development, AI-first design, software design and architecture, and high-performance teams. Jain received his Bachelor's in computer science engineering from VTU. Contact him at a.jain@devon.nl.

ROHIT TIWARI is a software architect at DevOn Software Services, Delft, 2612 CT, The Netherlands. His research interests include green software development and the applications of artificial intelligence in the software development lifecycle. Tiwari received his Bachelor's in computer science engineering from RTU. Contact him at r.tiwari@devon.nl.

This study explores the potential role of AI-driven tools across the software development lifecycle, suggesting their increasing adoption and possible impact on productivity. Survey insights underscore the value professionals place on these tools, especially for streamlining tasks in development, quality assurance, and DevOps, where they deliver notable efficiency and performance gains.

As AI technology evolves, its potential to reshape conventional workflows will only deepen. AI-driven tools not only assist developers but also equip organizations to meet higher standards of security, scalability, and maintainability. With ongoing advancements, these tools become increasingly adaptive, learning from project data to refine recommendations that align with changing team dynamics and project objectives. This adaptability makes AI tools invaluable for enterprises aiming to modernize practices, stay competitive, and create more value for customers.

It is essential, however, to recognize that the full impact of AI tools depends on specific use cases and the extent of integration into the SDLC workflow. While they offer significant productivity enhancements, there are challenges—such as data quality, bias, ethical concerns, and integration hurdles—that organizations must address. To maximize the benefits, this article emphasizes best practices like identifying the right use cases, focusing on tasks with high complexity and repeatability, and establishing quantifiable metrics to measure AI-driven productivity.

Looking ahead, the potential of AI in enterprise software development continues to grow, with trends suggesting a future where these tools become integral to agile, responsive, and efficient workflows. As enterprises prepare for this evolution, they must strategically balance innovation with ethical considerations and a commitment to responsible AI adoption to unlock the full potential of these transformative technologies. □

APPLYING AI TO SOFTWARE DEVELOPMENT

REFERENCES

1. "The complete guide to SDLC (Software development life cycle)." Atlassian. Accessed: Jan. 5, 2025. [Online]. Available: <https://www.atlassian.com/agile/software-development/sdlc>
2. D. Ajiga, P. A. Okeleke, S. O. Folorunsho, and C. Ezeigweneme, "Enhancing software development practices with AI insights in high-tech companies," *Comput. Sci. IT Res. J.*, vol. 5, no. 8, pp. 1897–1919, Aug. 2024, doi: [10.51594/csitrj.v5i8.1450](https://doi.org/10.51594/csitrj.v5i8.1450).
3. G. Naimil Navnit, "The future of software development: Integrating AI and machine learning into the SDLC," *Int. J. Eng. Manage. Res.*, vol. 14, pp. 308–315, Feb. 2024.
4. "Atlassian Intelligence accelerates your productivity on the Atlassian platform." Atlassian. Accessed: Nov. 12, 2024. [Online]. Available: <https://www.atlassian.com/platform/artificial-intelligence>
5. "The everything app for work." ClickUp. Accessed: Jan. 4, 2025. [Online]. Available: <https://clickup.com/>
6. "Project tracking software." Atlassian. Accessed: Jan. 4, 2025. [Online]. Available: <https://www.atlassian.com/software/jira>
7. "Manage your team's work, projects, & tasks online." Asana. Accessed: Jan. 4, 2025. [Online]. Available: <https://asana.com/>
8. "Trello brings all your tasks, teammates, and tools together." Trello. Accessed: Jan. 4, 2025. [Online]. Available: <https://trello.com/>
9. "The happier workspace." Notion. Accessed: Jan. 4, 2025. [Online]. Available: <https://www.notion.com/>
10. "How CI&T accelerated development by 11% with AI from Tabnine and Google Cloud." Tabnine. Accessed: Nov. 11, 2024. [Online]. Available: <https://www.tabnine.com/blog/how-cit-accelerated-development-by-11-with-ai-from-tabnine-and-google-cloud/>
11. Y. Raghavan. "The impact of Github copilot on developer productivity: A case study." Harness. Accessed: Nov. 11, 2024. [Online]. Available: <https://www.harness.io/blog/the-impact-of-github-copilot-on-developer-productivity-a-case-study>
12. "The AI editor for everyone." GitHub. Accessed: Dec. 30, 2024. [Online]. Available: <https://github.com/features/copilot>
13. "The AI software dev platform tailored to you - private, personalized, protected." Tabnine. Accessed: Jan. 4, 2025. [Online]. Available: <https://www.tabnine.com/>
14. "Amazon Q – generative AI assistant." Amazon Web Services. Accessed: Jan. 4, 2025. [Online]. Available: <https://aws.amazon.com/q/>
15. "Gemini code assist: An AI coding assistant." Google Cloud. Accessed: Jan. 4, 2025. [Online]. Available: <https://cloud.google.com/products/gemini/code-assist>
16. "Cursor - The AI code editor." Cursor AI. Accessed: Jan. 4, 2025. [Online]. Available: <https://www.cursor.com/>
17. M. Vijayvergiya et al., "AI-assisted assessment of coding practices in modern code review," in *Proc. 1st ACM Int. Conf. AI-Powered Softw. (AIware)*, New York, NY, USA: ACM, 2024, pp. 85–93, doi: [10.1145/3664646.3665664](https://doi.org/10.1145/3664646.3665664).
18. R. Zeeshan et al., "AI-powered code review with LLMs early results," 2024, arXiv:2404.18496.
19. E. Oliveira, S. Rios, and Z. Jiang, "AI-powered peer review process: An approach to enhance computer science students' engagement with code review in industry-based subjects," ASCILITE Publications. 2023. [Online]. Available: <https://publications.ascilite.org/index.php/APUB/article/view/482>
20. "CodeRabbit + Claude: AI-powered code review delivers 86 development." Anthropic. Accessed: Jan. 4, 2025. [Online]. Available: <https://www.anthropic.com/customers/coderabbit>
21. "Medrio was able to scale from 2 releases a year to 8 after implementing testRigor." testRigor AI-Based Automated Testing Tool. Accessed: Dec. 11, 2024. [Online]. Available: <https://testrigor.com/case-study-medrio/>
22. "Faster testing for your custom mobile, web, and Salesforce apps." Testim.io. Accessed: Jan. 4, 2025. [Online]. Available: <https://www.testim.io/>
23. "The AI-powered testing platform built for speed, scalability and accuracy." AppliTools. Accessed: Jan. 4, 2025. [Online]. Available: <https://applitools.com/>
24. "Create and run your tests faster at any scale." Katalon. Accessed: Jan. 4, 2025. [Online]. Available: <https://katalon.com/>
25. "Power your software testing with AI and cloud." LambdaTest. Accessed: Jan. 4, 2025. [Online]. Available: <https://www.lambdatest.com/>
26. "AI-powered codeless test automation on the cloud." ACCELQ. Accessed: Jan. 4, 2025. [Online]. Available: <https://www.accelq.com/>
27. "Mysa improves engineers productivity by 40% with harness." Harness. Accessed: Dec. 11, 2024. [Online]. Available: <https://www.harness.io/case-studies/mysa>
28. "DevOps for machine learning: Accelerating model development

- and deployment." TechBullion. Accessed: Jan. 4, 2025. [Online]. Available: <https://techbullion.com/devops-for-machine-learning-accelerating-model-development-and-deployment/>
29. S. Tatineni and A. Rodwal, "Leveraging AI for seamless integration of DevOps and MLOps: Techniques for automated testing, continuous delivery, and model governance," *J. Mach. Learn. Pharm. Res.*, vol. 2, no. 2, pp. 9–41, Sep. 2022.
30. P. Sami, U. Shah, N. Ahmad, and M. O. Beg, "Towards MLOps: A DevOps tools recommender system for machine learning system," 2024, *arXiv:2402.12867*.
31. Iresha Rubasinghe, D. Meedeniya, and I. Perera, *Tool Support for Software Artefact Traceability in DevOps Practice: SAT-Analyser*. Hershey, PA, USA: IGI Global, 2020.
32. "Modern monitoring and security." Datadog. Accessed: Jan. 4, 2025. [Online]. Available: <https://www.datadoghq.com/>
33. "Intelligent observability." New Relic. Accessed: Jan. 4, 2025. [Online]. Available: <https://newrelic.com/>
34. "The AI-native software delivery platform™." Harness. Accessed: Jan. 4, 2025. [Online]. Available: <https://www.harness.io/>
35. "Turn software delivery into a competitive advantage." Digital.ai. Accessed: Jan. 4, 2025. [Online]. Available: https://digital.ai/home-2/?utm_referrer=https%3A%2F%2Fwww.google.com%2F
36. "Unified observability and security." Dynatrace. Accessed: Jan. 4, 2025. [Online]. Available: <https://www.dynatrace.com/>
37. V. Shah, "Towards efficient software engineering in the era of AI and ML: Best practices and challenges," *Int. J. Comput. Sci. Technol.*, vol. 3, no. 3, pp. 63–78, 2019.
38. V. Dileep and P. D. N. K. Kommisetty, "Leading the future: Big data solutions, cloud migration, and AI-driven decision-making in modern enterprises," *Educ. Admin. Theory Pract. J.*, vol. 28, no. 3, pp. 352–364, 2022.
39. K. Nadeem and S. Aslam, Cloud-native DevOps strategies: Redefining enterprise architecture with artificial intelligence, 2024. [Online]. Available: https://www.researchgate.net/profile/Saleem-Aslam-9/publication/386071364_Cloud-Native_DevOps_Strategies_Redefining_Enterprise_Architecture_with_Artificial_Intelligence/links/6741ac5227661f7ae665deeb/Cloud-Native-DevOps-Strategies-Redefining-Enterprise-Ar
40. S. Rangaraju, S. Ness, and R. Dharmalingam, "Incorporating AI-driven strategies in DevSecOps for robust cloud security," *Int. J. Innovative Sci. Res. Technol.*, vol. 8, no. 11, pp. 2359–2365, 2023.
41. S. Kumari, Agile cloud transformation in enterprise systems: Integrating AI for continuous improvement, risk management, and scalability, *Aust. J. Mach. Learn. Res. Appl.*, vol. 2, no. 1, pp. 416–440, Mar. 2022.
42. A. Roy, "Measuring and quantifying cost, risk & value of AI initiatives." Gartner, Nov. 13, 2024. Accessed: Dec. 30, 2024. [Online]. Available: <https://www.gartner.com/en/newsroom/press-releases/2024-11-13-gartner-it-symposium-xpo-2024-kochi-india-day-3-highlights>
43. D. Ajiga, P. A. Okeleke, S. O. Folorunsho, and C. Ezeigweneme, "Navigating ethical considerations in software development and deployment in technological giants," *Int. J. Eng. Res. Updates*, vol. 7, no. 1, 2024, pp. 50–63, doi: [10.53430/ijeru.2024.7.1.0033](https://doi.org/10.53430/ijeru.2024.7.1.0033).
44. N. Rajkumar, C. Viji, A. Mohanraj, K. R. Senthilkumar, R. Jagajeevan, and J. A. Kovilpillai, "Ethical considerations of AI implementation in the library era," in *Improving Library Systems with AI: Applications, Approaches, and Bibliometric Insights*. Hershey, PA, USA: IGI Global, 2024, pp. 85–106.
45. R. Reddy Kethireddy, "Secure model distribution and deployment for LLMs," *J. Recent Trends Comput. Sci. Eng.*, vol. 12, no. 4, pp. 1–14, 2024, doi: [10.70589/JRTCSE.2024.4.1](https://doi.org/10.70589/JRTCSE.2024.4.1).
46. "Recommendation of the council on artificial intelligence." OECD Legal Instruments. Accessed: Jan. 5, 2025. [Online]. Available: <https://legalinstruments.oecd.org/en/instruments/oecd-legal-0449>
47. Information Technology — Artificial Intelligence — Artificial Intelligence Concepts and Terminology, ISO/IEC 22989:2022, 2022. Accessed: Jan. 5, 2025. [Online]. Available: <https://www.iso.org/standard/74296.html>
48. T. Sharma et al., "A survey on machine learning techniques applied to source code," *J. Syst. Softw.*, vol. 209, Mar. 2024, Art.no. 111934, doi: [10.1016/j.jss.2023.111934](https://doi.org/10.1016/j.jss.2023.111934).
49. A. Odeh, N. Odeh, and A. S. Mohammed, "A comparative review of AI techniques for automated code generation in software development: Advancements, challenges, and future directions," *TEM J.*, vol. 13, no. 1, 2024, pp. 726–739, doi: [10.18421/TEM131-76](https://doi.org/10.18421/TEM131-76).
50. Z. Du et al., "Multi-agent software development through cross-team collaboration," 2024, *arXiv:2406.08979*.
51. S. Hong et al., "MetaGPT: Meta programming for a multi-agent

- collaborative framework," 2024, *arXiv:2308.00352*.
52. W. Wysocki and C. Orłowski, "A multi-agent model for planning hybrid software processes," *Procedia Comput. Sci.*, vol. 159, pp. 1688–1697, Jan. 2019, doi: [10.1016/j.procs.2019.09.339](https://doi.org/10.1016/j.procs.2019.09.339).
53. P. Nama, "Integrating AI in testing automation: Enhancing test coverage and predictive analysis for improved software quality," *World J. Adv. Eng. Technol. Sci.*, vol. 13, no. 1, pp. 769–782, 2024, doi: [10.30574/wjaets.2024.13.1.0486](https://doi.org/10.30574/wjaets.2024.13.1.0486).
54. M. Pradel and S. Chandra, "Neural software analysis," *Commun. ACM*, vol. 65, no. 1, pp. 86–96, Dec. 2021, doi: [10.1145/3460348](https://doi.org/10.1145/3460348).
55. J. Pachouly, S. Ahirrao, K. Kotecha, G. Selvachandran, and A. Abraham, "A systematic literature review on software defect prediction using artificial intelligence: Datasets, data validation methods, approaches, and tools," *Eng. Appl. Artif. Intell.*, vol. 111, May 2022, Art.no. 104773, doi: [10.1016/j.engappai.2022.104773](https://doi.org/10.1016/j.engappai.2022.104773).
56. S. Kumari, "AI-powered agile project management for mobile product development: Enhancing time-to-market and feature delivery through machine learning and predictive analytics," *Afr. J. Artif. Intell. Sustain. Develop.*, vol. 3, no. 2, pp. 342–360, Dec. 2023.
57. A. Oyeniran, A. Adeleke, L. Akwawa, and P. Azubuko, "AI-driven DevOps: Leveraging machine learning for automated software deployment and maintenance," *Eng. Sci. Technol. J.*, vol. 4, no. 6, pp. 728–740, 2023.
58. W. Shah and A. Abbas, "DataOps meets Devops: AI-driven approaches for modernizing cloud enterprise architectures," 2021.
59. A. Jensen, "AI-driven DevOps: Enhancing automation with machine learning in AWS," *Integr. J. Sci. Technol.*, vol. 1, no. 2, Feb. 2024.
60. T. Venkata Mohit, "AI-powered continuous deployment: Leveraging machine learning for predictive monitoring and anomaly detection in DevOps environments," *Hong Kong J. AI Med.*, vol. 2, pp. 37–77, Feb. 2022.



CALL FOR ARTICLES

IT Professional seeks original submissions on technology solutions for the enterprise. Topics include

- emerging technologies,
- cloud computing,
- Web 2.0 and services,
- cybersecurity,
- mobile computing,
- green IT,
- RFID,
- social software,
- data management and mining,
- systems integration,
- communication networks,
- datacenter operations,
- IT asset management, and
- health information technology.

We welcome articles accompanied by web-based demos.
For more information, see our author guidelines at
www.computer.org/itpro/author.htm.

WWW.COMPUTER.ORG/ITPRO





Transforming Software Engineering Processes Through Generative AI: A Framework for Integration and Implementation

Aybuke Yalçiner^{ID}, TÜBİTAK, BİLGEM, Software Technologies Research Institute

Ebru Gökarp^{ID}, Hacettepe University

Ahmet Dikici^{ID}, ASELSAN, Inc.

In software engineering, integrating artificial intelligence (AI) technologies cuts costs and speeds up project timelines while improving workflow efficiency and software quality. We propose a framework that highlights tasks where leveraging generative AI can offer benefits through an analysis of current industry trends and research findings.

Generative artificial intelligence (AI) plays a role in the field of AI, focusing on generating unique content in different forms like text and visual media materials by using sophisticated models like Generative Pre-Trained Transformers (GPTs), Bidirectional Encoder Representations from Transformers

(BERTs), generative adversarial networks (GANs), and large language models (LLMs). These models can handle tasks beyond data processing operations. For example, they can write poetry, tell stories, create pieces of code, design beautiful art, and have conversations that seem real enough. They're good at it because they've been taught using a lot of different information and examples to understand patterns and language rules well. As a result, they can sometimes produce outputs that resemble human work.

Digital Object Identifier 10.1109/MC.2025.3539347
Date of current version: 27 June 2025

The evidence gathered from real-world experiences strongly confirms the influence of generative AI in the realm of software development. GitHub Copilot¹ stands out as an example. A generative AI tool can successfully help in coding by suggesting code snippets and answering coding queries efficiently during the development process. The efficiency gains are significant: tasks are completed 55% faster with Copilot in practice. According to research conducted with approximately 450 developers, Accenture found that using Copilot boosts team output and enhances the quality of work delivered.^a The positive impact extends to developer productivity well. Mercedes Benz study^b noted an increase of 30 min a day through leveraging this tool. Comparisons with traditional estimation methods reveal that GPT2SP² can deliver 34%-57% more accurate story point estimations for tasks within the same project. It is also likely to boast an impressive accuracy improvement of 39%-49% for multiproject estimations. Generative AI is now changing the face of software development by increasing productivity, providing correctness, and automating activities such as code reviews and security scans. It maintains the focus of developers and ensures early detection of quality and security. Moreover, it massively reduces time for bug fixes and test creation. This game-changing technology reduces the time required for bug fixes postlaunch by 50% through automation during test procedure creation. In addition, it enhances development speed: developers can write code 35%-45% faster and refactor it

20%-30% faster. Eliminating repetitive work from developers' activities results in more creative, meaningful work. In this regard, 87% of developers with access to generative AI can focus on engaging work, compared to 50% without it. This increases team creativity and encourages experimentation to provide a better user experience.³

The benefits of generative AI have attracted practitioners' and academics' attention. The current landscape of software engineering (SE) shows increasing integration of LLMs, particularly generative AI, which is one of the most emerging tools in SE. Very recently, Rajbhoj et al.⁴ identified successful applications in code generation, requirements gathering, and testing activities using such models. State-of-the-art studies show that LLMs speed up development processes and boost productivity while creating high-quality software artifacts.⁵ However, problems of quality, trustworthiness, and ethical consequences that result from AI-generated solutions have been identified as major challenges.⁶ Empirical studies reveal that generative AI is more deployed as an assistant collaborator than as a full automator, with effective interactions influenced by user context, among other factors, and organizational policies. In Khojah et al.,⁷ despite its potential, further research is needed to address scalability, integration with existing development practices, and the long-term impact of generative AI on software quality and team dynamics. Despite the growing body of research, there remains a significant gap in our understanding of how to integrate generative AI across SE processes comprehensively. To our knowledge, there is currently no study in the literature that outlines the areas

where generative AI can be applied in SE processes, discusses the challenges involved, presents a road map for successful implementation, and explains the potential future impacts and areas influenced by the widespread use of generative AI in SE.

This study aims to fill this gap by conducting a rigorous analysis to identify specific tasks within the SE processes defined in SWEBOK v4⁸ that can significantly benefit from the application of generative AI technologies.

This analysis is intended to provide SE professionals with actionable insights for effectively utilizing generative AI. It also aims to highlight the inherent limitations of the technology while exploring the promising possibilities it offers for future advancements.

The remainder of this article is organized as follows. The next section explores specific applications of generative AI in each SE process, providing detailed implementation examples. This is followed by a comprehensive road map for integrating generative AI into existing software development workflows, along with a discussion of potential challenges and practical solutions for organizations that adopt these technologies. Finally, emerging trends and future implications for the SE field are presented, concluding with a summary of the research findings.

INTEGRATION OF GENERATIVE AI

The developed framework, given in Table 1, was created by analyzing and synthesizing the findings of two significant systematic literature reviews^{9,10} conducted in 2024, one of which was conducted by the authors themselves.⁹

^a<https://www.youtube.com/watch?v=AAT4zCfzsHI>.
^b<https://www.youtube.com/watch?v=HdAqqNM1i9c>.

TABLE 1. Application of generative AI in SE processes.

SE process	Tools	Task	Explanation	Advantages of utilization of generative AI
Software requirements	ChatGPT, Claude AI, Requirements Assistant AI, IBM DOORS, and Google Bard	Anaphoric ambiguity treatment	Resolving ambiguities in natural language requirements to ensure clear interpretation	Enhancing clarity and precision by reducing interpretational uncertainties
		Requirements analysis and evaluation	Collecting, understanding, and documenting stakeholder needs and constraints	Accelerating requirements collection and enhancing documentation consistency
		Co-reference detection	Handling terminology variations across stakeholder documents over time	Eliminating inconsistencies and improving clarity and traceability
		Specification formalization	Translating requirements into precise, unambiguous formal language	Enhancing documentation precision, consistency, and interteam communication
		Use-case generation	Creating scenarios of system-user interactions	Improving system behavior understanding among developers and stakeholders
		Requirements classification	Categorizing requirements into functional and nonfunctional types	Developing a more effective software design road map
		Specification generation	Detailing requirement specifications	Ensuring correct solution development and facilitating stakeholder communication
		Requirements elicitation	Identifying software requirements	Enhancing understanding of software needs and reducing project failure risk
		Software specification synthesis	Automatically extracting requirements from natural language sources	Achieving high automation success, improving process efficiency and speed
Software architecture	ChatGPT, Claude AI, ArchiMate AI, Enterprise Architect AI, and Draw.io AI	System design	Creation of system architectures and components	Faster design iteration and reduced human error
Software design	ChatGPT, Claude AI, Figma AI, Sketch AI, Axure AI, and Zeplin AI	GUI retrieval	Finding and retrieving GUI designs or interfaces	Integration of text and visual information, and time and cost efficiency
		User design recommendation	Suggesting optimal user interface (UI) design	Improved user experiences and alignment with user needs
Software construction	ChatGPT, Claude AI, GitHub Copilot, TabNine, Sourcery AI, and SonarLint	Code representation	Encoding code semantics into distributed vector representations	Supporting downstream tasks like code search and generation
		Method name generation	Suggesting accurate method names	Improved code readability and understandability
		API documentation smells	Tracking API documentation quality	Enhanced developer experience
		Data analysis	Analyzing large datasets	Faster and more accurate analysis results
		Control flow graph generation	Illustrating program behavior	Supporting downstream tasks like code search and generation
		Instruction generation	Creating documentation from codebase, design, and requirements	Ensuring consistency, saving time, and improving accuracy

(Continued)

APPLYING AI TO SOFTWARE DEVELOPMENT

TABLE 1. (Continued.) Application of generative AI in SE processes.

SE process	Tools	Task	Explanation	Advantages of utilization of generative AI
		Code search	Extracting source code from large codebase	Enhanced code search and retrieval
		Code understanding	Deep analysis of code structure	Improved code maintenance and optimization
		Identifier normalization	Normalizing vocabulary in code snippets	Improving understanding and automation
		Type inference	Determining data types in code	Improving readability, simplifying maintenance, and minimizing runtime errors
		Code generation	Generating code snippets	Enhancing code-writing efficiency and accuracy
Software testing	ChatGPT, Claude AI, DeepCode, Selenium AI, Testim.io, and SonarQube AI	Vulnerability detection	Identifying system vulnerabilities	Improved detection accuracy and uncovering unknown vulnerabilities
		Testing automation	Assessing software application accuracy and reliability	Improved test coverage and reduced manual effort
		Defect detection	Identifying software defects	Enhanced bug detection and improved software reliability
		Static analysis	Analyzing source code	Improved accuracy and vulnerability identification
		Compiler fuzzing	Finding compiler vulnerabilities	Generating smart inputs and identifying deep compiler bugs
		Invariant prediction	Identifying critical consistent conditions	Enhanced testing and continuous improvement
		Test generation	Creating test cases	Identify test coverage gaps to ensure thorough testing
		Verification	Validating software correctness	Improved software reliability and security
		Fault localization	Identifying fault-triggering test cases	Increased probability of finding fault-inducing scenarios
		GUI testing	Verifying user interface (UI)	Faster, more comprehensive testing and improved UI consistency
		Decompilation	Translating compiled code to higher-level languages	Enhanced code understanding and faster reverse engineering
		Malicious code localization	Analyzing security-threatening code segments	Faster localization and reduced false positives
		Resource leak detection	Managing improper resource release	Optimized resource utilization and improved system performance
		Regression test prioritization	Organizing test-case execution order	Faster fault detection and efficient resource utilization
		Root-cause analysis	Identifying error origins	Improved problem solving and preventing error recurrence

(Continued)

TABLE 1. (Continued.) Application of generative AI in SE processes.

SE process	Tools	Task	Explanation	Advantages of utilization of generative AI
Software maintenance	ChatGPT, Claude AI, Snyk AI, Embold, Loggly AI, and Dynatrace	Story point estimation	Predicting task time and resources	Better task complexity assessment
		Software tool configuration	Optimizing tools for project needs	Reduced setup time and minimized errors
		Requirement implementation likelihood	Predicting requirement implementation probability	Improved resource allocation
		Cost estimation	Predicting required budget	Better budget control
		Delay estimation	Predicting potential project delays	Early bottleneck identification
		Code clone detection	Detects identical code snippets	Helps to improve code quality
		Debugging	Finding errors or defects in the software	Improves the accuracy and sampling efficiency of code generation and improves developer productivity
		Review/commit/code classification	Categorizes code changes and commits	Improves code-review efficiency and ensures high-quality, consistent code across the development process
		Logging	Systematic reporting of errors, info, warnings, and so on, and software application	Helps to identify and diagnose bugs and performance bottlenecks
		Code revision	Improvement of existing code	Helps with error detection and performance optimization, ensuring consistency
		Code coverage prediction	Identification of gaps in test coverage	Helps to improve test prioritization and enable efficient resource allocation and early defect detection
		Code-review defects repair	Identification of issues or defects found during code reviews	Speeds up code review, improves consistency, and helps to identify security risks, logic errors, and performance issues
		Docker file repair	Identification and resolution of issues in Docker file	Helps to improve the efficiency, security, and maintainability of Docker images
		Program merge conflicts repair	Identifies the issues that arise when integrating individual code changes	Resolves conflicts in a more efficient manner
		Tag recommendation	Recommends tags for software posts O&A sites	Better content organization and a more efficient, user-friendly platform
		Type error repair	Correction and detection of type error issues in code	Helps developers to resolve type errors quickly; more reliable software and improves developer productivity
		Maintainability of software likelihood	Predicting the long-term ease of maintaining and updating a software system as it evolves over time	Results in more efficient development processes over the long term, lowers maintenance costs, and enhances overall software quality

(Continued)

APPLYING AI TO SOFTWARE DEVELOPMENT

TABLE 1. (Continued.) Application of generative AI in SE processes.

SE process	Tools	Task	Explanation	Advantages of utilization of generative AI
SE management	ChatGPT, Claude AI, Monday.com AI, Jira AI, Trello AI, and Smartsheet AI	Story point estimation & Effort estimation	Prediction of time, resources and manpower to complete a specific task.	Better assessment of task complexity, improves project planning, and better decision making
		Software tool configuration	Software tools optimized for use according to project needs	Reduces setup time, minimizes human error, ensures that tools are consistently configured, and enhances overall productivity
		Requirement implementation likelihood	Prediction of requirement implementation probability	Helps teams to prioritize high-likelihood requirements, optimize resource allocation, and mitigate risks, leading to more successful project outcomes
		Cost estimation	Prediction of required budget to complete a specific task	Budget control, improves project planning.
		Delay estimation	Prediction of potential delays in a software development project	Identifying bottlenecks and risks early and ensures better project management
		Document changes prediction	Prediction of potential changes in the project documents	Reducing the risk of outdated or inconsistent information
		Delivery capability estimation	Prediction of a teams' ability to deliver a project within the given constraints	Helps teams to set realistic expectations, improve planning, and ensure on-time delivery

Insights from these reviews were used to systematically map generative AI applications across seven core SE processes defined in SWEBOK v4.⁸ Although these reviews offer valuable insights into the applications of generative AI in SE, there still exists a notable gap in our understanding of its comprehensive integration throughout the software development lifecycle. Insights from these reviews were examined and systematically mapped to generative AI applications across seven core SE processes defined in SWEBOK v4.⁸ The given study, while recognizing the importance of the other six knowledge areas described in the SWEBOK v4,⁸ focuses primarily on the following seven. The reason for this focus is that a narrower scope allows for deeper and more actionable insights into highly relevant domains in the current application of generative AI technologies. Additionally,

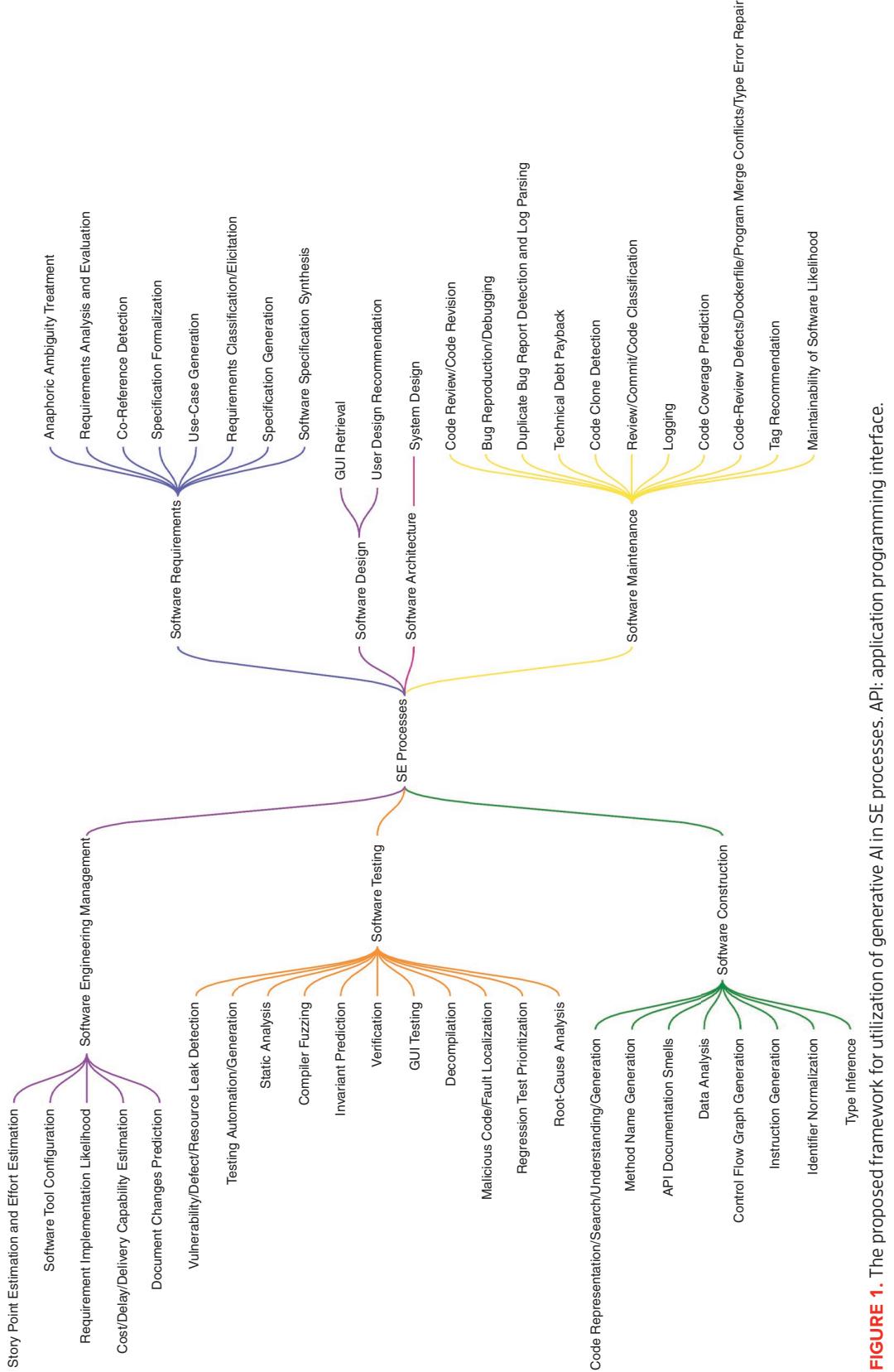
we aim to provide empirical evidence where the measurable impacts can be easily assessed, lay the groundwork for future research in the other knowledge areas, and highlight fields with high potential for industry adoption shortly. This mapping was conducted by three authors, each with more than 10 years of experience in academia and industry, working independently. Interrater reliability was assessed by using Cohen's kappa coefficient ($\kappa=0.85$), which indicates strong agreement. Any discrepancies were resolved through structured discussion sessions.

The framework identifies seven SE processes that are particularly conducive to generative AI integration: software requirements, software architecture, software design, software construction, software testing, software maintenance, and SE management, as illustrated in Figure 1.

Table 1 provides a detailed overview of the specific tasks within each process where generative AI can provide measurable benefits. It also outlines the potential advantages of leveraging generative AI for these tasks and recommends appropriate AI tools for each SE process, ensuring that teams can enhance their efficiency and effectiveness.

Software requirements

The success or failure of SE projects depends on accurately gathering and interpreting the requirements. Generative AI is a powerful tool for addressing these challenges by simplifying ambiguous language, resolving semantic ambiguities, creating detailed user-centered use cases, improving stakeholder communications, and aligning client expectations with the delivered project outcomes.



Software architecture

Architectural diagrams are necessary for reviewing and enhancing the long-term sustainability of a software system. This is enhanced with generative AI, where high-level design prototypes are built, analyzing existent codebases for ideal architectural patterns, identifying early vulnerabilities, and recom-

few developers consistently apply these principles when writing code. Generative AI transforms development by automating the creation of high-quality code snippets, providing real-time optimization suggestions, identifying potential bugs before implementation, streamlining debugging processes, and ensur-

ing security vulnerabilities, and reorganizing code. Proactive generative AI algorithms provide suggestions that turn maintenance into a proactive activity rather than a reactive one.

SE management

Effective SE management enables the delivery of projects on time, and within budget, while maintaining customer satisfaction and ensuring high-quality deliverables; it requires sophisticated resource coordination and predictive abilities. Generative AI will significantly enhance such practices by providing unparalleled management insights through better resource allocation, efficient usage of resources, proactive risk prediction, intelligent project scheduling, and accurate cost and effort estimation. It leverages insights from past project performance to enhance budgeting accuracy and enables real-time progress monitoring to predict on-time delivery. Generative AI combines historical project data with advanced predictive analytics, enabling managers to make data-driven decisions that improve the rate of project success.

Generative AI is transforming different aspects of SE, including agile software development,¹¹ by significantly enhancing how teams work and collaborate.¹² Mahboob et al.¹³ discuss the role of generative AI in different agile software development approaches, such as XP or Lean, and agile practices, such as pair programming, test-first development, continuous integration, and refactoring, to enhance the productivity of agile teams. This can be highly beneficial for Scrum teams, as generative AI serves as a valuable assistant for key roles. For Scrum masters, this provides strategic event-facilitation

STATE-OF-THE-ART STUDIES SHOW THAT LLMs SPEED UP DEVELOPMENT PROCESSES AND BOOST PRODUCTIVITY WHILE CREATING HIGH-QUALITY SOFTWARE ARTIFACTS.

mending best practices based on historical data. Ultimately, this improves the efficiency and reliability of software architecture.

Software design

In the software design phase, generative AI uses several machine learning methods, such as LLMs with natural language processing, to create detailed design models that help to optimize design decisions and improve the overall design process for functional and nonfunctional system requirements more accurately and quickly.

Software construction

Code quality is essential for the success of other phases in the software lifecycle. Well-constructed, efficient, and readable code simplifies testing and maintenance, especially during changes in project requirements, making refactoring easier and less error-prone. Although much has been written about coding best practices,

ing consistent coding standards with greater agility.

Software testing

Testing is an important quality control mechanism that defines performance benchmarks and reduces potential issues. This is how generative AI will completely change testing: by automatically generating test cases, intelligently identifying bugs, anticipating error conditions, continuously learning from historical test results, and reducing manual testing efforts. Generative AI continuously improves testing techniques, raising software reliability and user satisfaction to higher levels.

Software maintenance

Maintenance is a resource-intensive yet very crucial software development lifecycle phase. Generative AI improves the process by detecting code inefficiencies, recommending targeted improvements, automating dependency updates, identifying potential

suggestions, presents insightful performance metrics, and detects bottlenecks in the workflow effectively. Similarly, for a product owner, this streamlines backlog prioritization, refines user stories, assists in market analysis, and offers predictive analytics.¹¹ Especially generative AI is effective in supporting agile practices like pair programming, test-driven development, continuous integration, and refactoring. Many studies have shown that the integration of generative AI into Scrum processes leads to significant improvements in team efficiency, innovation potential, and the quality of decision making, along with accelerating software delivery timelines. Embracing this technology will facilitate agile teams to get to higher levels of effectiveness and success that could not have been reached before.

IMPLEMENTATION OF GENERATIVE AI

Major integrational challenges of generative AI into SE processes, among others, include issues on integrations to current

SE tools and workflows, high model complexities, lack of interpretability, very costly computations and implementations, ethics concerns, and responsibility for the inclusion of AI. The most particularly felt complexities of the models do include an algorithmic burden when it comes to special operations of the AI in given tasks considered during SE.¹⁴ Despite these challenges, the benefits of adopting generative AI into SE processes are significant. Software organizations are eager to adopt it, but careful planning and execution are crucial to fully realize its long-term benefits. Otherwise, a poorly managed implementation can result in complex and unmanageable systems. To ensure success, it is essential to follow a structured road map and focus on the key areas that support AI adoption and effective data management.¹⁵ The implementation steps are illustrated in Figure 2.

Initial assessment and goal definition

The first phase focuses on assessment and goal setting. Organizations

must evaluate potential high-impact areas including requirements engineering, code generation, testing, and security, while defining clear objectives and success metrics for their implementation.

Determine how to manage data

The second phase is vital for establishing a robust data management framework, which is critical for the success of any organization. The development of holistic data storage and access protocols makes sure that our data are secure and of high quality. This captures accurate depictions of real-world scenarios and further refines the relevance and efficacy of generative AI models. Organizations must now implement metadata tracking and governance systems while enabling for capabilities of real-time data integration. A well-considered data management strategy forms the backbone of successful generative AI adoption. Training of generative AI models requires high-quality, accurate, and relevant data to ensure the optimal

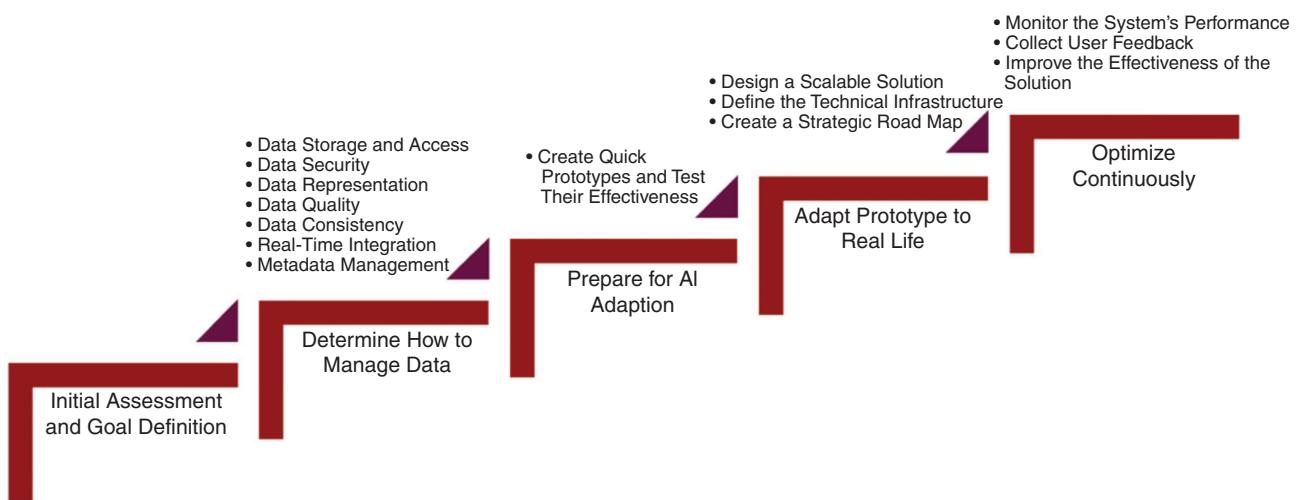


FIGURE 2. The steps to be followed for implementing generative AI.

results. This mainly concerns the determination of how and where the data will be stored and who will have access to them. Stringent data security should be implemented to ensure that sensitive information does not leak into the hands of unauthorized persons. It is also important to balance data to reflect real-life situations to increase the reliability of the model significantly. Implementing rigorous processes for data quality and consistency is crucial to maintaining accuracy across systems. Real-time data integration will make our AI models dynamic and responsive, thus best positioned to adapt to changing needs. Finally, effective metadata management enables lineage tracking and enhances governance for accountability and traceability. If all these critical areas are addressed, organizations are in a better position to provide a strong foundational framework for generative AI, thereby helping to ensure the fantastic performance of SE processes.

Prepare for AI adaption

Tool selection and prototyping is an important third phase, taking from two to eight weeks. At this stage, organizations make a strategic selection of the appropriate AI tools like GPT-4, Codex, or Copilot for organizational need analysis. Teams design rapid prototypes for the most important use cases, ensuring that these are in line with larger business objectives. The resources required are also identified in the process, while test applications are tried out under real-world conditions. Moreover, it is essential to assess the data and resources that are required for each use case as well as any gaps or technical challenges. This prototyping allows for extensive testing of the practical applications of

AI and also forms an important step in the validation of concepts before full-scale implementation, therefore laying the grounds for success.

Adapt prototype to real life

Production implementation is the fourth phase and takes two to four months. During the first two to four weeks, organizations develop scalable infrastructure and devise an implementation road map. At this stage, full deployment across business units occurs as the prototype goes live in a production system. This includes extensive testing of all components of the system, coupled with intensive team training. To perfect the prototype into a fully operational generative AI system, precise fine-tuning ensures its reliability and effectiveness. Additionally, internal teams should receive proper training to ensure smooth assimilation and optimal usage of the new system. This concludes with the capability to make necessary changes and achieve successful large-scale implementation, laying a foundation for organizational success in general.

Optimize continuously

Continuous optimization is key in the final stage to ensure that the system remains effective and efficient over time. An organization should continuously monitor performance metrics, elicit user feedback actively, and scale data pipelines accordingly when required. Organizations can effectively meet emerging demands by enhancing system integration and strengthening security measures.

At this stage, all the modifications in the data pipeline must adapt to handle the increase in the volume of information. Moreover, integrating

with other business systems and processes allows organizations to introduce new functionality in response to user needs and feedback. In addition, improving security protocols is essential for protecting sensitive data; it is also important to develop tools for continuous performance monitoring and optimization. Ethical considerations need to be addressed. Organizations have to be sure that their operation complies with emerging regulations and involves responsible behavior in practice. Building ethical guidance into the use of generative AI concerning data privacy, decision making, and security will enhance integrity while building trust. As companies need to comply with evolving regulatory frameworks, generative AI models must be transparent, explainable, and free from biases. As generative AI continues to surge into prominence, a commitment to organizational readiness for adoption and compliance will be necessary to realize long-term success and enable responsible deployment.

CHALLENGES OF GENERATIVE AI

Although this advanced technology can greatly speed up and enhance the development of solutions, which increases efficiency overall, it has to be engaged quite cautiously, given the presence of prominent risks. Organizations need to intelligently work their way around any mishaps that can lead to compromising project outcomes. These challenges include, as listed in **Table 2**, professional and ethical issues beyond technical aspects. Addressing them effectively demands strategic and sensitive stewardship so as not to incur unforeseen adverse

impacts. To realize the full power of generative AI, organizations should engage in a holistic approach to balance between technological innovation and human expertise. It means appropriate framing for responsible AI adoption, ensuring that such powerful tools augment and do not replace human intelligence, that is, developing adaptive implementation strategies with the key premises of transparency, continuous learning, and ethical considerations. Positioning generative AI more as a collaborative partner and less as a pure substitute for human developers allows organizations to realize its transformational potential while mitigating many of the associated risks. The dynamic and thought-provoking presentation needs to place generative AI in its proper context: it is no more than an advanced tool designed to augment rather than replace human creativity and problem-solving skills.¹⁶

FUTURE TRENDS

The future of generative AI in SE represents a significant transformation that could impact numerous aspects of the field.^{17,18} We foresee a radical shift from traditional, experience-based approaches to more sophisticated data-driven decision-making processes as generative AI becomes even more pervasive within development workflows. This transition makes basic activities like design, testing, and deployment faster and more accurate, which results in faster and more dependable software delivery.

Generative AI in the field of DevSecOps will play an important role in security by offering early vulnerability detection and remediation across the development lifecycle. This proactive approach to risk mitigation will lead to

higher application resilience. In addition, software development platforms are becoming easier and more intuitive to use, and development is possible with little to no formal coding knowledge among users. Technology democratization will become a key trend, with software development platforms becoming more accessible. Nontraditional coders will be better equipped to build apps, extending the innovation lifecycle and diversifying the talent pool. It should broaden access and eliminate traditional barriers to entry in software development. The role of cloud-based solutions will continue to increase, given their highly scalable and flexible nature. As the maturity of cloud computing advances, organizations will be more agile and quick to adapt to market dynamics and technological shifts. The convergence of cloud technologies

with generative AI will make software infrastructures more responsive and intelligent. Quantum computing represents another frontier that will challenge current programming paradigms, requiring new approaches to software development. Developers will need to continuously evolve their skills, embracing new tools and techniques that leverage quantum computational capabilities. This technological disruption will push the boundaries of computational limits. Ethical considerations and evolving regulatory frameworks will increasingly shape technological development. New programming languages and development practices will emerge, prioritizing performance, simplicity, and robust security features. These advancements will not only redefine SE but also create diverse career opportunities for emerging talent.

TABLE 2. Challenges of generative AI.

Challenge	Explanation
Risks of overreliance on AI-generated code	Developers may assume that AI-generated code is inherently reliable, but such assumptions can lead to security vulnerabilities and errors, compromising software quality and project integrity.
Data privacy and security risks in AI training	Training AI systems requires large datasets, which may inadvertently expose sensitive project data, creating significant risks to confidentiality and security.
Financial implications of AI system development	The development and implementation of generative AI models can be resource intensive, involving high costs for model optimization, training, and efficient data handling.
Bias and adaptability challenges in AI models	Poorly balanced training datasets result in biased AI outputs, and with the evolution of SE, AI models have to be constantly updated to keep them effective, which adds complexity and costs.
Impact on developer creativity and problem solving	Overreliance on generative AI can lead to the loss of creativity and critical thinking among developers because they could stop developing original solutions and just stick with what the AI model suggests.
Concerns over job security in the age of AI	The widespread adoption of generative AI raises a wide range of concerns among developers regarding job security as AI would potentially replace human roles and make the profession yet uncertain.

ABOUT THE AUTHORS

AYBÜKE YALÇINER is a senior researcher at the Software Technologies Research Institute (YTE), The Informatics and Information Security Research Center (BILGEM), Scientific and Technological Research Council of Turkey (TUBITAK), Ankara 06530, Turkey. Her research interest includes data driven software engineering. Yalçiner received her master's degree in computer science from Hacettepe University. Contact her at aybukeyalciner@hacettepe.edu.tr.

EBRU GÖKALP is an associate professor in the Computer Engineering department at Hacettepe University, Ankara 06800, Turkey. Her research interests include software engineering, information systems, and Industry 4.0. Gökalp received her Ph.D. degree in information systems from Middle East Technical University. Contact her at ebrugokalp@hacettepe.edu.tr.

AHMET DİKICI is a senior chief team leader at ASELSAN Inc., Ankara 06200, Turkey. His research interests include software engineering, business process management, and software process improvement. Dikici received his Ph.D. degree in information systems from Middle East Technical University. Contact him at ahmetdikici@aselstan.com.

growing amount of research on human-AI collaboration, this study builds on initial efforts by delving deeply into a nuanced perspective of how generative AI could augment human decision making and productivity throughout the software development lifecycle. By addressing challenges, providing a clear implementation road map, and exploring future implications, we have established a practical framework that is applicable to general SE processes. However, it is crucial to recognize the needs of different industries, especially those that require high precision and sound logical reasoning, such as aerospace, medical device manufacturing, financial trading, traffic management, and scientific computing, necessitating special studies.

A primary limitation of this study is our focused examination of seven out of the thirteen SWEBOK v4⁸ knowledge areas. This selective approach was driven by the need for deep, actionable analysis within reasonable study parameters, and the current technological readiness of different areas for AI integration. Future research should address the remaining six areas. Particularly promising directions include investigating how generative AI could enhance SE processes and methods, support professional practice through automated guidance, and optimize engineering economics through predictive modeling. This sequential approach will allow us to track technological readiness and practical applicability as generative AI capabilities mature. Future studies also need to investigate not only the current role and limitations of generative AI within such specialized contexts but also how such roles might evolve as the technology matures. For instance, in aerospace engineering, generative AI might initially serve as a

Our thorough investigation into the role of generative AI in SE uncovered a transformative technological shift that holds the potential to fundamentally reshape the development processes outlined in SWEBOK v4.⁸ By examining the application of generative AI across essential SE processes, from requirements gathering to maintenance, we have identified significant opportunities to improve productivity, efficiency, and creativity. On the other hand, the success of generative AI is a function of both the quality of the underlying AI models and how effectively users can leverage those models. For instance, even the most state-of-the-art model might still produce poor results if not leveraged properly, while even a moderately advanced model

could perform extremely well if placed in the hands of an expert.

The key to maximizing the benefit of generative AI is balance: leveraging technological innovation with human expertise. An organization needs to plan well and commit to transparency and ethical practices. Generative AI is designed to support human intelligence, not replace it; hence, strategies to enhance creativity and problem solving need to be instituted. In return, this allows organizations to maximize the benefits of generative AI, reducing the risk of potential misuse that includes misinformation and the invasion of privacy. Understanding generative AI as an enhancer of human capabilities is central to its responsible and effective integration.¹⁹ Although there is a

documentation assistant before evolving into a more sophisticated verification tool; in web development, it might progress from code completion to full-stack architectural recommendations. Understanding these evolutionary pathways and domain-specific adaptations will be key to the effective integration of generative AI across diverse contexts in SE.

As we look to the future, generative AI is poised to transform software development. As AI technology advances, its capacity to automate complex tasks, enhance decision making, and drive innovation will continue to expand. To tackle this situation, software professionals must cultivate agility and adaptability. These advances are not a threat but a powerful tool for continuous improvement. The future of SE will be characterized by a collaborative relationship between human creativity and AI to lead and innovate in an increasingly digital world. □

REFERENCES

1. D. Smit and H. Smuts, "The impact of GitHub Copilot on developer productivity from a software engineering body of knowledge perspective," in *Proc. Americas Conf. Inf. Syst.*, 2024, pp. 1-10.
2. M. Fu and C. Tantithamthavorn, "GPT2SP: A transformer-based agile story point estimation approach," *IEEE Trans. Softw. Eng.*, vol. 49, no. 2, pp. 611–625, Feb. 2023, doi: [10.1109/TSE.2022.3158252](https://doi.org/10.1109/TSE.2022.3158252).
3. R. Seroter, "Finding flow: Generative AI's impact on developer productivity," Google Cloud. Accessed: 2024. [Online]. Available: URL link: <https://cloud.google.com/resources/gen-ai-impact-on-dev-productivity>
4. A. Rajbhoj, A. Somase, P. Kulkarni, and V. Kulkarni, "Accelerating software development using generative AI: ChatGPT case study," in *Proc. 17th Innov. Softw. Eng. Conf. (ISEC)*, New York, NY, USA: ACM, pp. 1-11, 2024, doi: [10.1145/3641399.3641403](https://doi.org/10.1145/3641399.3641403).
5. Y. Huang et al., "Generative software engineering," 2024, *arXiv:2403.02583*.
6. J. Sauvola, S. Tarkoma, M. Klemettilin, J. Riekki, and D. Doermann, "Future of software development with generative AI," *Automated Softw. Eng.*, vol. 31, no. 1, 2024, Art. no. 26, doi: [10.1007/s10515-024-00426-z](https://doi.org/10.1007/s10515-024-00426-z).
7. R. Khojah, M. Mohamad, P. Leitner, F. G. Neto, and O. de, "Beyond code generation: An observational study of ChatGPT usage in software engineering practice," 2024, *arXiv:2404.14901*.
8. H. Washizaki, "Guide to the software engineering body of knowledge (SWEBOK guide), Version 4.0." IEEE Computer Society, Waseda Univ., Tokyo, Japan, 2024. [Online]. Available: <http://www.swebok.org>
9. A. Yalçınler, A. Dikici, and E. Gökalp, "Data-driven software engineering: A systematic literature review," in *Systems, Software and Services Process Improvement*, M. Yilmaz, P. Clarke, A. Riel, R. Messnarz, C. Greiner, and T. Peisl, Cham, Switzerland: Springer-Verlag, 2024, pp. 19-32, doi: [10.1007/978-3-031-71139-8](https://doi.org/10.1007/978-3-031-71139-8).
10. X. Hou et al., "Large language models for software engineering: A systematic literature review," *ACM Trans. Software Eng. Method.*, vol. 33, no. 8, pp. 1-79, 2024, doi: [10.1145/3695988](https://doi.org/10.1145/3695988).
11. E. Naiburg, "AI as a scrum team member," *Scrum.org*, Jul. 10, 2024. Accessed: Nov. 19, 2024. [Online]. Available: URL link: <https://www.scrum.org/resources/blog/ai-scrum-team-member>
12. J. W. Beard et al., "Agile development: The promise, the reality, the opportunity," in *Proc. Agil-ISE@CAiSE*, 2024, pp. 7-17.
13. M. Mahboob, M. Rayyan Uddin Ahmed, Z. Zia, M. Shakeel Ali, and A. Khaleel Ahmed, "Future of artificial intelligence in agile software development," 2024, *arXiv:2408.00703*.
14. U. K. Durrani, M. Akpinar, M. F. Adak, A. T. Kabakus, M. M. Ozturk, and M. Saleh, "A decade of progress: A systematic literature review on the integration of AI in software engineering phases and activities (2013-2023)," *IEEE Access*, vol. 12, pp. 171,185-171,204, 2024, doi: [10.1109/ACCESS.2024.3488904](https://doi.org/10.1109/ACCESS.2024.3488904).
15. Y. Mota, "Generative AI implementation: Comprehensive guide," Oct. 15, 2024. Accessed: Nov. 19, 2024. [Online]. Available: <https://www.n-ix.com/generative-ai-implementation/>
16. K. Wach et al., "The dark side of generative artificial intelligence: A critical analysis of controversies and risks of ChatGPT," *Entrepreneurial Bus. Econ. Rev.*, vol. 11, no. 2, pp. 7-30, 2023, doi: [10.15678/EBER.2023.110201](https://doi.org/10.15678/EBER.2023.110201).
17. R. Jha, "Future of software development with generative AI & machine learning," *Autom. Softw. Eng.*, vol. 31, no. 1, Sep. 2024, Art.no. 26.
18. P. Kumari, "7 top generative AI use case in software engineering," *Labellerr*, Feb. 12, 2024. Accessed: Nov. 19, 2024. [Online]. Available: <https://www.labellerr.com/blog/the-most-elaborative-guide-for-generative-ai-with-top-7-use-cases/#:~:text=Generative%20AI%20has%20diverse%20applications,%2C%20summarization%2C%20and%20dialogue%20generation>
19. C. Ebert and P. Louridas, "Generative AI for software practitioners," *IEEE Softw.*, vol. 40, no. 4, pp. 30-38, Aug. 2023, doi: [10.1109/MS.2023.3265877](https://doi.org/10.1109/MS.2023.3265877).



ChatGPT Choreography: Discovering Developer Dialogues and Potential Software Development Lifecycle Applications

Aman Swaraj, Sandeep Kumar[✉], and Lalit Mohan Sharma, Indian Institute of Technology Roorkee

We analyze a large dataset of 17913 ChatGPT–developer dialogues using topic modeling and thematic analysis. Our multimethod approach identifies 10 themes, revealing key insights into ChatGPT’s application across software development lifecycle tasks and the time required for effective issue resolution.

Artificial intelligence (AI) has become an invaluable resource for developers today. From automating repetitive tasks to enabling complex data analysis, AI has transformed the way software is conceived, developed, and deployed.^a

Within this context, generative AI tools, such as ChatGPT, have further emerged as powerful assets in generating human-like responses to real-time queries. Exceeding 1 million users within five days of its launch,

it now (as of September 2024) receives more than a billion visits every month on average.^b

Despite this widespread adoption of ChatGPT, there is limited research on how developers integrate it into their workflows. However, addressing this research gap is vital for the software community.

By analyzing the dynamics of developer prompts and ChatGPT’s proposed solutions, we can provide actionable insights for both developers and tool designers. For developers, this understanding can optimize problem-solving strategies, improve workflow efficiency, and offer guidance on when and how to leverage AI tools across various stages of the software development

^a<https://github.blog/news-insights/research/survey-reveals-ais-impact-on-the-developer-experience/>

lifecycle (SDLC). Similarly, for tool designers and platform administrators, these insights can guide improvements in future iterations of AI tools, refining response accuracy and tailoring functionalities to better meet developer needs. Moreover, this also aligns with previous research concerning developer discussions and technical trends on open source platforms, such as Stack Overflow (SO) and GitHub,^{1,2} highlighting the value of studying these interactions.

In this context, the recently released public dataset DevGPT,³ comprising 17,913 ChatGPT-developer prompts, offers a valuable resource. This dataset contains a variety of software development artifacts, such as GitHub commits, pull requests, issues, and discussions, making it ideal for analyzing ChatGPT's role in real-world software development scenarios.

While some researchers have analyzed this dataset, their methods had notable limitations. For instance, Champa et al.⁴ employed a zero-shot classification technique to categorize the prompts into predefined software engineering themes. However, this method may not be ideal for tasks involving highly technical or context-specific content. Since this method assumes that all conversations fit within established labels, it can lead to the neglect of emerging patterns or unique discussions. Moreover, there is a risk that conversations that do not clearly match any predefined label may still be classified into one with low confidence. This was evident in Champa et al.'s study, where no threshold was set for zero-shot classification probabilities, leading to prompts being clustered into predefined labels even with low probability scores. Additionally, their validation set consisted of only 20 prompts, a

very small sample size, further limiting the reliability of their findings.

In contrast, several works focused on qualitative approaches to categorize the conversations. For instance, Hao et al.⁵ performed a manual investigation of 580 conversations from the DevGPT dataset. However, their analysis was limited to conversations related to GitHub pull requests and issues, not covering the remaining portion of the dataset. In a similar attempt, Siddiq et al.⁶ applied an open coding method to analyze the DevGPT dataset from the pull request category. However, their sample set was limited to 39 conversations only. Similarly, Mohamed et al.⁷ and Sadgic et al.⁸ also adopted the open coding approach to identify different themes in the DevGPT dataset. However, their work also faced challenges similar to scalability and subjectivity bias.

While these studies offered valuable insights, they were limited by their sample sizes and specific focus, often examining only a small portion of the dataset. This issue of small sample size can largely be attributed to the inherent constraints of manual analysis. Further, in addition to the scalability challenges, relying solely on manual investigation can also introduce biases that can affect the overall findings.

To address these limitations, we introduce a hybrid approach combining quantitative techniques, such as BERTopic modeling,⁹ with a structured thematic analysis. BERTopic modeling is an unsupervised topic modeling technique that leverages BERT embeddings to identify and cluster topics within text data, making it particularly effective for uncovering patterns in large, complex datasets. On the other hand, thematic analysis allows for a detailed examination of

qualitative data, helping to identify themes and insights that emerge from developer interactions.

Specifically, in this work we employ a four-stage thematic analysis combining both inductive and deductive coding methods, ensuring more comprehensive exploration than manual annotation alone. This rigorous approach reduces subjectivity through its structured process, while enhancing scalability by focusing manual efforts on refining and validating automated topics.

Overall, our hybrid approach combining BERTopic modeling and thematic analysis enables a more comprehensive discovery of the content presented to ChatGPT by developers, addressing the limitations of previous works. Table 1 presents a comparative analysis of our study with the prior works conducted in this domain.

Additionally, we also aim to examine the time taken by ChatGPT (number of turns) to resolve the queries raised by the developers across all of the explored themes. This analysis can be crucial as it can offer useful insights into user behavior, the complexity of issues, and the effectiveness of ChatGPT in addressing those issues.

Another aspect of these conversations that hasn't been explored in previous studies involves predicting the chat length based on the initial query made by the user. This problem statement first appeared as a research question in the Mining Software Repositories Challenge (MSR'24),^c highlighting its significance within the research community.

We reason that addressing this question can provide several benefits

^c<https://2024.msrconf.org/track/msr-2024-mining-challenge/#Call-for-Mining-Challenge-Papers>

TABLE 1. Comparative analysis of developer–ChatGPT interaction studies.

Study	Methodology	Data size	Limitations
Champa et al. ⁴	Zero-shot classification	2,865 prompts	No threshold for classification; small validation set; lacks qualitative analysis
Hao et al. ⁵	Manual investigation	580 prompts	Limited to GitHub pull requests and issues; lacks automated analysis; subjectivity bias and scalability issues
Siddiq et al. ⁶	Open coding	39 prompts	Small sample size; lacks automated analysis; subjectivity bias and scalability issues
Mohamet et al. ⁷	Open coding	512 prompts	Subjectivity bias and scalability issues; lacks automated analysis
Sagdic et al. ⁸	Manual annotation	1,701 prompts	Subjectivity bias and scalability issues
Our study	BERTopic and hybrid thematic analysis	2,865 prompts	—

to the software community. For developers, it can assist in better time management as the tool can inform them in advance about the estimated duration of a conversation. This can allow the developers to plan their coding sessions, adjust their queries, or seek alternative solutions, especially when working under tight schedules. Further, it can be useful at various stages of the SDLC, such as during testing, where iterative debugging may require significant time, or maintenance, where quick resolutions are critical. Additionally, for the AI service providers, predicting conversation length can aid in optimizing server load and resource allocation, ensuring smoother operation during peak times and enhancing user experience.

Based on these observations and the gaps in the existing literature, our study seeks to address the following research questions (RQs):

- » RQ1: What are the primary themes in software engineering where developers seek assistance from ChatGPT?

- » RQ2: What is the average time taken by ChatGPT (number of turns) to provide solutions for queries under these identified themes?
- » RQ3: How accurately can we predict the length of a ChatGPT conversation based on the initial query provided by the user?

To address RQ1, we conduct an in-depth analysis of the DevGPT dataset using the aforementioned hybrid approach. Building on these findings, for RQ2, we apply zero-shot classification to assign conversations to the newly defined themes and analyze response times by calculating the average number of turns per category. Finally, for RQ3, we generate context-dependent embeddings using multiple variants of the sentence-transformer framework, which are effective for capturing nuanced semantic relationships. Once the embeddings are generated, we experiment with multiple regression-based models for predicting the conversation length.

Our main contribution is the in-depth analysis of the DevGPT

dataset, presenting a taxonomy of 10 themes where developers seek assistance from ChatGPT, revealing key usage trends and preferences. Additionally, we incorporate insights from broader surveys and studies on generative AI tools in SDLC workflows, suggesting areas for improvement in future models to better support developers and optimize AI-driven assistance.

METHODOLOGY OVERVIEW

Our study workflow, as illustrated in Figure 1, involves topic discovery, thematic analysis, average turn per theme calculation, and the working of the chat length prediction model.

TOPIC DISCOVERY

Since the dataset was originally in the form of links, we began by using the Selenium^d library to download the dataset in HTML format. Then, we utilized BeautifulSoup^e to extract the text and save it into CSV files.

^d<https://pypi.org/project/selenium/>

^e<https://pypi.org/project/beautifulsoup4/>

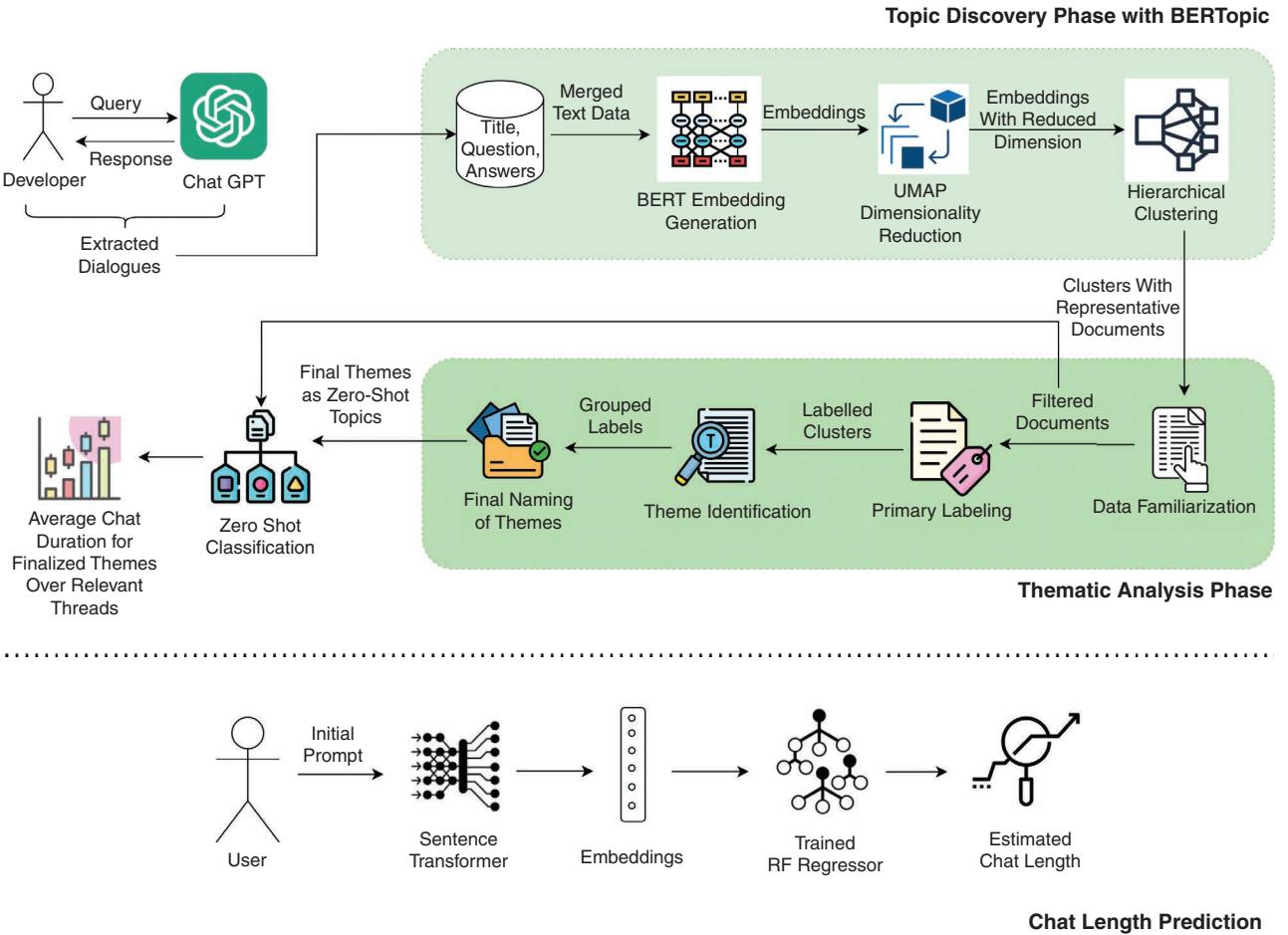


FIGURE 1. Overall workflow of our approach for topic discovery and chat length prediction.

After data extraction, we applied minimal data-cleaning steps and started the topic discovery process. We utilized BERTopic for our work, which is a state-of-the-art technique in topic modeling.

Unlike previous studies that relied heavily on manual investigation, our BERTopic approach automates the topic discovery process, providing a distinct advantage in both efficiency and coverage. BERTopic operates by first embedding text data into high-dimensional vectors and subsequently clustering them based on similarity.

This method facilitates the automatic discovery of topics without the need for predefined labels. We selected the “all-MiniLM-L6-v2” model because it offers performance comparable to the “all-mpnet-base-v2” model while being more computationally efficient due to its smaller parameter size.¹⁰

Additionally, it allows for the visualization of selected terms for each topic using bar charts derived from “c-TF-IDF” scores. Since these visualizations provide insights into the relative importance of terms, they help in

getting better comparisons between topic representations.

After applying BERTopic, a total of 87 clusters were generated from the dataset. We display some of the relevant topics with their corresponding word scores in [Figure 2\(a\)](#). Although BERTopic was previously applied in a related study,⁸ the potential benefit was not fully realized, as the authors manually reviewed all documents within each discovered cluster. In contrast, we refined our analysis by selecting the top 10 representative documents from each cluster, producing

a manageable 870 conversations. This filtration allowed us to capture the core of each topic, as BERTopic's hierarchical structure makes the top-ranked documents indicative of the overall cluster.

THEMATIC ORGANIZATION

Once the initial clusters were generated, the next step involved organizing the discovered topic clusters into meaningful categories. For this purpose, we used

a hybrid thematic analysis approach as given by Fereday and Muir-Cochrane,¹¹ combining inductive and deductive coding. This particular approach is suited for our study as it allows us to



FIGURE 2. Topic word scores. Some of the (a) relevant topics and (b) noncontextual topics along with their word scores identified by BERTopic analysis.

integrate data-driven insights with theoretical frameworks, ensuring a comprehensive understanding of the developer interactions. Our method comprised four stages: data familiarization, primary labeling, theme identification with review, and final naming.

Familiarization with data

We started by examining the representative documents and keywords associated with each of the 87 clusters. This process involved reading through the documents to understand the context and terminologies used in the interactions. We also consulted relevant forums and resources to clarify any unfamiliar terms. After familiarizing ourselves with the data, we filtered out 13 clusters that were not related to our domain. This left us with 74 relevant clusters for further analysis. Figure 2(b) displays some of the noncontextual topics identified in the corpus.

Primary labeling

In this stage, we assigned a descriptive label to summarize the core idea expressed within the documents for each of the remaining clusters. This step involved multiple rounds of review and refinement to ensure that the labels accurately represented the content of the clusters. The main goal in this phase was to capture the essence of each cluster in a concise sentence.

Theme identification and review

To refine the 74 labeled clusters into cohesive themes, we progressively grouped related clusters based on shared characteristics. Earlier in the data familiarization phase, we discarded some clusters based on their word score charts as they were clearly irrelevant to our focus [Figure 2(b)]. However, for the remaining clusters,

we conducted a detailed examination of our initial set of 10 representative documents from each cluster.

Here, we observed that although BERTopic clustered these documents by similarity, nuanced differences emerged in 22 clusters. Some clusters included overlapping themes, while others contained a mixture of relevant and irrelevant content. To address this, we even reviewed additional representative documents beyond the initial 10 where it was necessary, ensuring a thorough exploration of each cluster's content. Through this process, we formed 14 preliminary themes encompassing clusters with similar content, leaving some clusters as potential outliers if they did not fit perfectly within any theme. However, these outliers were retained as part of the exploratory nature of the study and noted within the closest matching themes to ensure no data was discarded unnecessarily. This gradual merging approach enabled us to retain all relevant information while focusing on core themes.

Final naming

In the final stage, we refined and named the themes to represent the patterns identified. Through further analysis, we consolidated the 14 themes into a final set of 10 primary themes, incorporating less frequent or unique clusters as subtopics or outliers within these themes. This approach allowed us to capture the full breadth of developer interactions with ChatGPT without excluding any relevant information.

AVERAGE TURN COUNT PER THEME

After performing thematic analysis, we classified the relevant conversations

using zero-shot classification, a machine learning method where a pretrained model assigns labels to unseen data points by understanding their underlying semantics.

In our context, we used the finalized themes from the previous step as the candidate labels. We used the model "facebook/bart-large-mnli" for classifying the samples as it is well suited for natural language tasks¹² and has been employed in a similar work.⁴ Unlike prior work, however, we applied this model selectively to relevant threads rather than the entire dataset, with predefined labels specifically tailored from our dataset's thematic analysis.

After selecting the zero-shot topics, the model calculated the probability of each sample fitting within these topics and assigned the label with the highest probability score, followed by the calculation of the average turns for all of the themes.

CHAT LENGTH PREDICTION

In connection with our RQ3, the first step involved extracting semantic features from the text-based data. For this purpose, we utilized the sentence-transformer framework to generate context-dependent embeddings for the user prompts. We experimented with three leading variants, namely "all-MiniLM-L6-v2", "BAAI/bge-m3", and "hkunlp/instructor-xl", which respectively comprise 384, 1,024, and 768-dimensional embeddings.

Next, to predict chat length from embeddings, we used some well-known regression models: random forest, support vector regressor (SVR), and gradient boosting (GradB).

We split the dataset into 80% training, 10% validation, and 10% testing to ensure sufficient training data and reduce

chances of overfitting. Next, for hyperparameter tuning, we applied grid search with a predefined parameter grid and fivefold cross-validation on the training set, optimizing for negative root mean squared error (RMSE). The grid included variations in `n_estimators`, `max_depth`, `min_samples_split`, and `min_samples_leaf` to systematically identify the best-performing configuration.

Using the embeddings as input features, we trained regression models to predict chat length. All models were evaluated using mean squared error (MSE), RMSE, and mean absolute error (MAE). These metrics are commonly used and well-established in regression tasks.

For training the model, we primarily focused on analyzing the initial prompt as it serves as a concise representation of the developer's intent. Further, it makes the approach computationally efficient and practical for real-time predictions. However, understanding the value of early-turn interactions for additional context, we conducted two sets of experiments: one using only the initial prompt and another using the first two prompts.

All experiments were conducted in Python language on Jupyter notebook on a system with Intel(R) Xeon(R) CPU E5-1650 v3 @ 3.50 GHz, 32 GB RAM, NVIDIA Quadro K620 GPU (2GB).

FINDINGS OF OUR RESEARCH QUESTIONS

The 10 finalized themes, along with descriptions, example prompts, the number of conversations, and average chat lengths, are outlined in Tables 2–4. While these 10 categories represent broad software engineering tasks, specific SDLC stages are also addressed within these broader themes. We expand on these themes in the following sections, highlighting specific

engagement patterns. Further, we integrate insights from other studies and surveys to contextualize our findings and discuss their SDLC implications in the “*Insights From Broader Surveys and Studies*” and “*Key SDLC Takeaways and Future Opportunities*” sections.

In addition, we also provide a set of developer prompts related to SDLC stages in [Table 4](#), with an extended version available on our GitHub repository.^f

Software engineering-specific tasks and usage patterns

- **Code generation:** This is the most common request with 416 conversations, averaging 5.12 turns each. Developers frequently use ChatGPT for snippets in specific languages or for converting code across languages, emphasizing the tool's utility for quick code generation needs.
- **Issue and error resolution:** Many discussions revolve around troubleshooting with ChatGPT, as developers share error messages and code snippets. The category “issue and error resolution” included 281 conversations, indicating that developers heavily rely on ChatGPT for assistance with testing. Furthermore, these conversations had an average of 5.05 turns each, reflecting the iterative process through which ChatGPT addresses these error-handling requests.
- **Conceptual clarification:** Developers frequently seek high-level explanations on topics, ranging from memory management (e.g., WebAssembly limits) to software design patterns.

- **How-to guidance:** Developers also frequently request step-by-step guidance for tasks, including compatibility issues and framework setup.
- **Review and refactoring:** Code improvement and refactoring requests are also common, with developers seeking feedback on implementation choices.

Patterns by specific domains and functional areas

- **Front-end and back-end development:** This category is represented in “web development” (206 conversations with an average of 8.74 turns), showing a more extended exchange, likely due to the complexity and breadth of web development queries. Developers working with frameworks like React and Spring Boot use ChatGPT for setup and troubleshooting.
- **DevOps:** Developers request assistance with continuous integration/continuous delivery (CI/CD), containerization (e.g., Docker), and version control, often leaning on ChatGPT for recommendations and best practices.
- **Data management:** In “data handling” (245 conversations, averaging 6.39 turns), developers commonly interact with ChatGPT for structured query language queries, data analysis, and optimization tasks, with a high number of turns indicative of detailed interactions.
- **AI and machine learning (ML):** Developers use ChatGPT for model building and understanding ML/DL libraries, with the AI and ML category averaging 4.27 turns, suggesting shorter

^fhttps://github.com/Anonymous-sdlcovid/ChatGPT_Choreography

engagements but frequent usage of ChatGPT for queries requiring AI insights.

Additional functional areas

➤ *Translation and text editing:* A significant portion of chats comprised text editing and translation tasks, highlighting developers' frequent reliance on ChatGPT

for documentation needs and utilitarian tasks. Together, these themes (222 and 267 conversations, respectively) indicate a high demand for assistance in refining documentation, translating technical content, and formatting text, demonstrating its utility in maintaining and updating technical documentation.

- **Optimization:** This category (averaging 6.57 turns) reflects developers' focus on improving code efficiency, optimizing resource utilization, and structuring efficient memory management.
- **Security:** Developers seek guidance on security practices, from token generation to encryption strategies. The "security"

TABLE 2. Summary of results for RQ1 and RQ2: Analysis of developer–ChatGPT conversations

Topic	Conversations related to	Example initial prompt	No. of chats	Average turns
AI and ML	AI and ML techniques, algorithms, models, and applications.	What are the most used activation functions in deep learning?	166	4.27
Coding assistance	Programming languages and coding practices.	Implement a ranged random float function in C++.	416	5.12
Data handling	Data management, analysis, and visualization techniques.	Structured query language query for fetching records, find the highest run scorer batsman from every team and his runs.	245	6.39
Development tools	Software development tools, frameworks, libraries, integrated development environments, etc.	What tools can I use to develop a Jupyter notebook freely? Please provide me the necessary references.	372	4.17
Issue and error resolving	Software bugs, errors, issues, and troubleshooting strategies.	How do I fix this Python error: no module named bs4?	281	5.05
Optimization	Performance optimization, code efficiency, algorithmic complexity, resource utilization.	Is it possible to optimize memory usage of a struct in Golang by ordering the properties on memory boundaries? How can I go about optimizing my structs for memory usage?	162	6.57
Security	Cybersecurity, data privacy, encryption, authentication, etc.	Hi, I have this class for generating user tokens in my ACL system using System.Text in ACL Net Core. Could you help with security and cryptography?	138	4.77
Text editing	Text generation, manipulation, formatting, and editing.	As a technical editor, you are an expert in grammar and plain language. Provide guidance on the following usage...	267	5
Translation	Translation services, language localization, and multilingual support in software applications.	I'm writing technical documentation in Markdown. Can you help translate it from Chinese to English?	222	5.82
Web development	Web development technologies, frameworks, libraries, front-end and back-end development.	What is the best way to change the page title when using React?	206	8.74

TABLE 3. Summary of results for RQ3: Performance of regression models for chat length prediction.

Model	Embedding	MSE		RMSE		MAE	
		Initial	Two prompts	Initial	Two prompts	Initial	Two prompts
RF	all-MiniLM	7.5	7.2	2.74	2.68	1.1	1.05
	bge-m3	6	5.8	2.45	2.39	0.93	0.89
	instructor-xl	8	7.9	2.83	2.81	1.2	1.15
SVR	all-MiniLM	8.7	8.4	2.95	2.9	1.3	1.25
	bge-m3	6.9	6.7	2.63	2.6	1.05	1
	instructor-xl	9.1	8.8	3.02	2.98	1.4	1.35
GradB	all-MiniLM	9.2	9	3.03	3	1.35	1.3
	bge-m3	7.2	7	2.68	2.65	1.1	1.05
	instructor-xl	9.5	9.3	3.08	3.05	1.5	1.45

Boldface entries represent the best performing model and text embedding combination for our proposed chat-length prediction model.

category, with an average chat length of 4.77 turns, shows stable engagement as developers look to integrate sound security practices into their code.

In conclusion, our analysis of RQ1 and RQ2 shows ChatGPT's wide-ranging utility across developer tasks, with specific patterns in engagement length and frequency based on task complexity.

Regarding RQ3, the BGE model achieved the lowest MAE of 0.93 ($R^2 = 91.78\%$) owing to its optimized structure, utilizing dense retrieval and multigranular embeddings up to 1,024 dimensions. However, other models also achieved low error values showing the efficacy of transformer models at predicting ChatGPT chat lengths based on user queries.

Similarly, among the regression models, the random forest regressor stood out for its robustness in handling complex data patterns effectively. Results of our RQ3 are also outlined in Table 3, along with findings of RQ1 and RQ2 in Table 2.

Including the second prompt slightly reduced the error, but the improvement was minimal. This can be attributed to the fact that the DevGPT dataset contained about 40% single-turn conversations, while 47% had two turns. As a result, the training data didn't have enough variation to show significant improvement with two prompts. While further increasing the number of turns may improve results, it would also incur higher computational costs. Moving further, exploration into specific query types, complexities, and error classes could hold promising future work.

INSIGHTS FROM BROADER SURVEYS AND STUDIES

Besides our analysis of the DevGPT dataset, we also aimed to incorporate insights from other relevant surveys and studies to contextualize our findings. To achieve this, we selected the SO Developers survey (2024),^g a valuable resource for understanding developers' preferences and usage patterns. Additionally, we studied publicly available datasets, specifically analyzing developer interactions with GitHub Copilot.^{h,i} Finally, we considered broader studies covering multiple AI tools to provide a more comprehensive perspective.^{13,14} Together, these resources offer valuable insights into developer interactions with AI tools and their usage patterns. The key findings from these studies are summarized in Table 5.

From a comparative viewpoint, these studies align with our analysis of the DevGPT dataset in several aspects. We see that coding and implementation emerged as the most frequent use case, emphasizing developers' reliance on AI tools for code generation and debugging. Furthermore, from the SO survey, we see that developers remain skeptical about using AI tools for complex tasks. This pattern can also be seen in our analysis, where optimization-related prompts required longer interactions (6.57 turns). Similarly, testing and deployment tasks were consistently identified as challenging across studies, which can account for the lesser presence of testing-related prompts in the DevGPT dataset.

Building on these observations, we outline key opportunities to enhance ChatGPT's support across different stages of SDLC in the next section.

^g<https://survey.stackoverflow.co/2024/>

^h<https://doi.org/10.5281/zenodo.7604509>

ⁱ<https://doi.org/10.5281/zenodo.7713789>

TABLE 4. Examples of conversations across SDLC phases.

SDLC phase	Example prompts
Documentation	<ul style="list-style-type: none"> Good evening ChatGPT, I'd like your help to write a README for using the bioinformatics tool GNINA tool on the Plex platform by LabDAO. Act as an enthusiast developer advocate with 5 years of experience. Write a quick documentation about this release Bash script...
Implementation	<ul style="list-style-type: none"> We're in the process of creating a Terraform provider for the Storyblok CMS. Through that, we can leverage infrastructure as code to manage the CMS's configuration. I'm creating an Nginx-like web server in C++98. The instructions say I have to give the option to turn on or off directory listing. What is this, and how can I implement it?
Design	<ul style="list-style-type: none"> When asking if a user is enjoying your app, is it common practice to open up a review window if they say yes? I'm a Ruby on Rails developer using version 7. By default, there are three environments: test, development, and production. I would like to add an integration environment. What would be the recommended way?
Planning and requirement gathering	<ul style="list-style-type: none"> I've recently experimented with Firebase, and I wonder how much time it saves in app development compared to a more traditional design. You are a product manager receiving requests from a client for building some software. You have to ensure that the requirements are reasonable and suited to solving the problem the users have.
Deployment and maintenance	<ul style="list-style-type: none"> Let's say we are working on the deploy of a web application. We'll start by defining our meta language notation and its meanings. I develop a local application called ActivityWatch that runs an API on localhost:5600. The API is only meant for local use in a web UI hosted from the same web server...I want an analysis and ways to address the issue.
Testing	<ul style="list-style-type: none"> I need to write a test for Cypress where I'm testing uploading a torrent file to a website with a multipart form. I need a test paragraph for a piece of software I'm working on (25 words or less).

API: application programming interface; UI: user interface.

KEY SDLC TAKEAWAYS AND FUTURE OPPORTUNITIES

➤ **Enhanced code assistance:** ChatGPT is heavily used for code generation and debugging, highlighting its role in the implementation phase. Future improvements could focus on language-specific optimizations, enhanced contextual understanding of ongoing projects, and better integration with code repositories and CI/CD pipelines to enable smooth and context-aware assistance.

➤ **Task estimation and planning insights:** As per the SO survey, developers rarely use AI tools for planning related tasks. Given our RQ3 analysis for

predicting conversation length, future large language models could potentially estimate task duration based on the nature of the prompt. This enhancement could provide users with estimated completion times, guiding them on how deep an interaction might be and setting user expectations. This could further provide developers with time estimates for various SDLC stages, from initial planning through debugging and testing, helping them manage project timelines effectively.

➤ **Intelligent support for testing:** Similar to planning, testing also remains an area of lower confidence for AI tools. Future

work could focus on enhancing ChatGPT's ability to generate reliable test cases, validate edge scenarios, and identify vulnerabilities. Integrating AI tools with well-established test automation frameworks (for example, Selenium, JUnit, PyTest) could be a practical step in this direction.

➤ **Documentation support:** Although ChatGPT is frequently used for text editing, translation, and documentation support, concerns remain about the lack of compliance with nonfunctional requirements and alignment with evolving project contexts. Enhancing ChatGPT to integrate with version control systems for documentation (for example, Git-based

APPLYING AI TO SOFTWARE DEVELOPMENT

TABLE 5. Insights from other relevant surveys and studies on developer–AI tool interaction.

Study	Insights on developer–AI usage patterns
Stack Overflow Development Survey (2024) ^g	<ul style="list-style-type: none"> Seventy-six percent of developers are either employing or planning to employ AI tools in their workflow. The most common use cases include “Writing code (82%),” “Searching for answers (67.5%),” “Debugging (56.7%),” and “Documentation (40%).” While currently, developers are skeptical about using AI tools for testing purposes, they are eager to integrate it in their workflows in the coming future. On the other hand, developers rarely use AI assistants for “Deployment (4.5%)” and “Planning (12.2%)” related works. Around 45% of developers using AI tools find them incompetent in handling complex tasks. ChatGPT is the most preferred AI tool among developers (82% usage), followed by GitHub Copilot (41%) and Google Gemini (24%).
Interaction with GitHub Copilot ^{h,i}	<ul style="list-style-type: none"> Interactions with programming assistants typically follow two modes: acceleration and exploration. In acceleration mode, the programmer knows their goal and uses tools like Copilot to reach it more efficiently. In exploration mode, the programmer is uncertain about the next steps and relies on Copilot to navigate potential solutions. The primary programming languages used by developers while working with Copilot are JavaScript and Python, with Visual Studio Code being the most common integrated development environment. The most frequently used technology is Node.js, and the leading functionality implemented is data processing. While the tool’s significant benefit is efficient code generation, a notable limitation is the difficulty of integration into existing workflows.
Other studies on AI–developer interaction ^{13,14}	<ul style="list-style-type: none"> Key benefits and use cases: Developers primarily use AI tools for autocompletion, reducing keystrokes, speeding up tasks, and recalling syntax. These tools excel in repetitive code, simple logic, quality assurance, and documentation, but are less effective for complex reasoning, brainstorming, and testing. Motivations: Developers are primarily motivated to use AI tools for autocompletion (86%), faster task completion (76%), avoiding online syntax searches (68%), finding solutions (50%), and identifying edge cases (36%). Barriers: On the other hand, some major barriers include unmet requirements (54%), difficulty controlling tools (48%), and unhelpful suggestions (34%), along with time-consuming debugging (38%) and poor contextual understanding. Adoption trends: 84.2% of developers use AI tools occasionally or regularly, and the most popular one is ChatGPT (72.1%). Other tools include GitHub Copilot (37.9%), JetBrains AI Assistant (28.9%), and Visual Studio IntelliCode (17.5%). A key reason developers prefer ChatGPT is its flexibility in supporting natural language interactions. Overall opinions: While developers appreciate the overall usability of AI assistants, security of the AI-generated code remains a primary concern for them along with lack of compliance to nonfunctional requirements. Common SDLC stage and functions: The most common stage for using AI tools is implementing new features (87.3%), with popular functions being feature generation and summarization. However, developers look forward to delegating testing and artifact generation tasks to AI tools as well.

workflows) could improve traceability and collaboration across development teams. Additionally, enabling ChatGPT to pull context-aware information from official documentation and developer forums could also improve the workflow. This feature would also be useful in the maintenance phase, where accurate and

updated documentation is essential for knowledge transfer.

➤ **Ethics and security:** Finally, security concerns and trust issues regarding AI-generated outputs also remain one of the critical aspects. To address these issues, future work could explore integrating ethical practices, such as giving proper attribution to

the most relevant and trusted sources and maintaining feedback loops for continuous model improvement. A significant step in this direction has been initiated by the SO community, as outlined in their guidelines.^j

^g<https://stackoverflow.blog/2024/02/29/defining-socially-responsible-ai-how-we-select-api-partners/>

By listing out these opportunities, we lay the foundation for future work aimed at making ChatGPT more context-aware, reliable, and aligned with the SDLC.

THREATS TO VALIDITY

The first threat in our work on analyzing ChatGPT-developer conversations could arise from the DevGPT dataset itself, as a single dataset cannot exhaustively represent the variety of discussion topics between developers and ChatGPT. While this is a notable constraint, the DevGPT dataset has been used extensively in existing works and serves as the benchmark in this domain. To complement this, we have included insights from broader surveys, such as the SO Developer survey, and studies focused on GitHub Copilot.

The presence of overlapping content in the finalized themes is another potential threat, which has been acknowledged in earlier studies as well. To address this concern, we employed a rigorous hybrid thematic analysis approach, incorporating both inductive and deductive coding, to ensure the clarity and distinctiveness of our categorization.

As the adoption of generative AI tools like ChatGPT continues to rise in the software community, understanding specific developer use cases becomes crucial. In this study, we have addressed this concern by analyzing a large set of ChatGPT-developer conversations from the DevGPT dataset. Using BER-Topic modeling and thematic analysis, we identified 10 primary themes, each with distinct interaction patterns that reveal ChatGPT's varied effectiveness across software engineering tasks. We also developed a regression model

ABOUT THE AUTHORS

AMAN SWARAJ is a research scholar in the Department of Computer Science and Engineering at the Indian Institute of Technology Roorkee, 247667 Roorkee, India. His research interests include natural language processing (large language models), computer vision, and human-centered studies. Swaraj received an M.Tech in computer science and engineering from the National Institute of Technology, Delhi. Contact him at aman_s@cs.iitr.ac.in.

SANDEEP KUMAR is a professor in the Department of Computer Science and Engineering, Indian Institute of Technology Roorkee, 247667 Roorkee, India. His research interests include web services, software engineering, and artificial intelligence. Kumar received a Ph.D. in computer engineering from the Indian Institute of Technology - BHU, Varanasi, India. He is a Senior Member of IEEE. Contact him at sandeep.garg@cs.iitr.ac.in.

LALIT MOHAN SHARMA is a student at the Indian Institute of Technology Roorkee, 247667 Roorkee, India. His research interests include machine learning, web development, and blockchain. Sharma is pursuing an undergraduate degree from Indian Institute of Technology Roorkee and is involved in project work in information security and software engineering laboratory. Contact him at lalitmohan@gmail.com.

using sentence transformers to predict chat lengths, achieving a mean absolute error of 0.93 turns. Our findings offer actionable insights into optimizing ChatGPT for developers and suggest several directions for future large language model enhancements based on observed usage trends. 

REFERENCES

1. R. Abdalkareem, E. Shihab, and J. Rilling, "What do developers use the crowd for? A study using stack overflow," *IEEE Softw.*, vol. 34, no. 2, pp. 53–60, Mar./Apr. 2017, doi: [10.1109/MS.2017.31](https://doi.org/10.1109/MS.2017.31).
2. A. Barua, S. W. Thomas, and A. E. Hassan, "What are developers talking about? An analysis of topics and trends in stack overflow," *Empirical Softw. Eng.*, vol. 19, no. 3, pp. 619–654, 2014, doi: [10.1007/s10664-012-9231-y](https://doi.org/10.1007/s10664-012-9231-y).
3. T. Xiao, C. Treude, H. Hata, and K. Matsumoto, "DevGPT: Studying developer-ChatGPT conversations," 2023, *arXiv:2309.03914*.
4. A. I. Champa, M. F. Rabbi, C. Nachuma, and M. F. Zibran, "ChatGPT in action: Analyzing its use in software development," in *Proc. 21st Int. Conf. Mining Softw. Repositories*, 2024, pp. 182–186, doi: [10.1145/3643991.3645077](https://doi.org/10.1145/3643991.3645077).
5. H. Hao et al., "An empirical study on developers shared conversations with ChatGPT in GitHub pull requests and issues," 2024, *arXiv:2403.10468*.

APPLYING AI TO SOFTWARE DEVELOPMENT

6. M. Latif Siddiq, L. Roney, J. Zhang, and J. C. S. Santos, "Quality assessment of ChatGPT generated code and their use by developers," in Proc. 21st Int. Conf. Mining Softw. Repositories, Mining Challenge Track (MSR), New York, NY, USA: Association for Computing Machinery, 2024, p. 6, doi: [10.1145/3643991.3645071](https://doi.org/10.1145/3643991.3645071).
7. S. Mohamed, A. Parvin, and E. Parra, "Chatting with AI: Deciphering developer conversations with ChatGPT," in Proc. IEEE/ACM 21st Int. Conf. Mining Softw. Repositories (MSR), Piscataway, NJ, USA: IEEE Press, 2024, pp. 187–191, doi: [10.1145/3643991.3645078](https://doi.org/10.1145/3643991.3645078).
8. E. Sagdic, A. Bayram, and M. R. Islam, "On the taxonomy of developers'
- discussion topics with ChatGPT," in Proc. IEEE/ACM 21st Int. Conf. Mining Softw. Repositories (MSR), Piscataway, NJ, USA: IEEE Press, 2024, pp. 197–201.
9. M. Grootendorst, "BERTopic: Neural topic modeling with a class-based TF-IDF procedure," 2022, *arXiv:2203.05794*.
10. Y. He, Z. Yuan, J. Chen, and I. Horrocks, "Language models as hierarchy encoders," 2024, *arXiv:2401.11374*.
11. J. Fereday and E. Muir-Cochrane, "Demonstrating rigor using thematic analysis: A hybrid approach of inductive and deductive coding and theme development," *Int. J. Qualitative Methods*, vol. 5, no. 1, pp. 80–92, 2006, doi: [10.1177/160940690600500107](https://doi.org/10.1177/160940690600500107).
12. M. Lewis et al., "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," 2019, *arXiv:1910.13461*.
13. A. Sergeyuk et al., "Using AI-based coding assistants in practice: State of affairs, perceptions, and ways forward," *Inf. Softw. Technol.*, vol. 178, Feb. 2025, Art. no. 107610, doi: [10.1016/j.infsof.2024.107610](https://doi.org/10.1016/j.infsof.2024.107610).
14. J. T. Liang, C. Yang, and B. A. Myers, "A large-scale survey on the usability of AI programming assistants: Successes and challenges," in Proc. 46th IEEE/ACM Int. Conf. Softw. Eng., 2024, pp. 1–13, doi: [10.1145/3597503.3608128](https://doi.org/10.1145/3597503.3608128).



IEEE Security & Privacy magazine provides articles with both a practical and research bent by the top thinkers in the field.

- stay current on the latest security tools and theories and gain invaluable practical and research knowledge,
- learn more about the latest techniques and cutting-edge technology, and
- discover case studies, tutorials, columns, and in-depth interviews and podcasts for the information security industry.



computer.org/security





Leveraging LLMs for Trustworthy Software Engineering: Insights and Challenges

Marco Vieira¹, University of North Carolina at Charlotte

Large language models (LLMs) are transforming software engineering by accelerating development, reducing complexity, and cutting costs. If fully integrated into the software lifecycle they will have the potential to drive design, development, and deployment. However, LLM-driven trustworthy software engineering requires addressing multiple challenges.

Large language models (LLMs) are revolutionizing how we engage with technology, from improving natural language processing to automating tasks and supporting decision making. These models are trained on extensive datasets and demonstrate great capabilities in analyzing and generating context-specific content.¹

With software deeply embedded into our daily lives, from critical power grids and health-care systems to

financial networks and transportation, trustworthiness is becoming essential.² A key challenge is managing complexity while maintaining quality attributes such as reliability, security, scalability, fairness, and ease of maintenance. This is increasingly difficult as software integrates a wider range of technologies [cloud computing, microservices, artificial intelligence/machine learning (AI/ML) components, edge devices, etc.] and agile development and continuous integration and continuous deployment (CI/CD) demand faster development cycles.

Developers need to deal with the balance between fostering innovation, rigorously testing code, preserving

legacy systems, and addressing ethical concerns regarding privacy and AI/ML. LLMs offer promising support to build trustworthy software systems in this context. Such models can be leveraged to improve key processes across the development lifecycle, from requirements elicitation and architecture design to code generation, testing, and issue management, among others.

In code generation, LLMs can support developers in producing higher-quality code by adhering to best coding practices in a time- and cost-effective manner.³ LLMs have also been studied for their ability to support architectural decisions.⁴ Furthermore, LLMs have the potential to automate code analysis and testing by identifying code patterns and generating comprehensive test cases, which can significantly reduce the cost of fixing issues later in the development process.⁵ In addition, LLMs can contribute to improving issue management by analyzing bug reports, prioritizing security vulnerabilities, and supporting root cause analysis.⁶

LLMs are already being used in software development for isolated tasks, but a holistic vision that considers broader trustworthiness objectives is missing. Our vision consists of integrating LLMs across the entire software development lifecycle (SDLC), from requirements gathering to the deployment of infrastructure as code (IaC) and post-deployment monitoring, allowing development teams to continuously improve their systems. This integration should be guided by the principle of trustworthiness throughout the entire software lifecycle—a perspective that, to the best of our knowledge, has not been extensively discussed before.

To explore the potential of LLMs in trustworthy software engineering, we consider the following two key questions:

1. How can LLMs be used across the SDLC to enhance trustworthiness (including both current and potential future applications)?
2. What are the challenges and risks of using LLMs in trustworthy software engineering, and what are the high-level directions for their mitigation?

We recognize that the full integration of LLMs in the SDLC (and other ideas discussed in this article) may currently be seen as speculative as the full applicability of LLMs to software engineering remains an open question, but this is aligned with our intent to explore not only the current state but also the future potential of LLMs in this domain.

While this article primarily focuses on the role of LLMs in enabling trustworthy software engineering, it is important to acknowledge that project management practices such as effort estimation, backlog organization, and sprint planning are critical components of the SDLC. These practices play a key role in aligning development efforts with timelines, resource constraints, and stakeholder priorities. Although this work does not discuss the integration of LLMs with project management processes, their potential in this domain is also promising. For instance, LLMs could potentially assist in effort estimation by analyzing historical project data and task complexities or optimize backlog organization by prioritizing tasks based on factors such

as dependencies, risk, and customer impact.

The remainder of this article is organized as follows. First, we frame the problem by examining the concepts of trust and trustworthiness. We then explore the role of LLMs in trustworthy software development. Following this, we discuss key challenges along with high-level strategies for their mitigation. Finally, we summarize the key aspects and potential directions for research.

TRUST AND TRUSTWORTHINESS

Trust and trustworthiness have been extensively studied across various domains. Numerous works in the literature focus on the issue of trust in social relationships as well as trust and trustworthiness within business environments.⁷ Although trust is defined differently across various fields, a common objective in all definitions is the precise assessment of trust levels, which serve as a foundation for informed decision making. Thus, establishing trustworthiness is both the first and most essential step in fostering trust.

In the software context, trust can be understood as a stakeholder's reliance on a system to behave as expected.⁸ This reliance is inherently risky as it is often based on a subjective belief formed through past experiences with the same or other systems. Consequently, the trust level can be interpreted as the estimated probability of such reliance, which is uncertain and subject to dynamic change. In other words, trustworthiness can be defined as the degree to which a software system deserves to be trusted.

Trustworthiness is a key concern for developers, researchers, and enterprises.^{8,9,10} However, several

factors contribute to the difficulty of ensuring trustworthiness. These include the diversity of software systems, the scale and complexity of modern systems, and the subjective nature of trust and trustworthiness. Depending on the context (for example, critical or noncritical systems), different quality attributes (for example, security or performance) may play a role in the system's trustworthiness.

Trustworthy software requires a range of functional and nonfunctional requirements to be met. While the attributes and metrics used to evaluate software functionality can vary depending on the specific purpose, they remain independent of the environment in which the software operates. The relative importance of nonfunctional requirements, however, depends on factors such as the criticality of the software (for example, safety critical or business critical), the significance of the data handled (for example, private data), and financial implications (for example, monetary transactions), among many others.

In short, common trustworthiness requirements include the following⁹:

- **Security:** Protect data and operations from unauthorized access or breaches, ensuring confidentiality, integrity, and availability.
 - **Reliability:** Perform functions accurately and consistently.
 - **Privacy:** User data must be handled responsibly, complying with regulations.
 - **Robustness:** Handle unexpected inputs or stress conditions gracefully.
 - **Maintainability:** Code should be designed for ease of updates and modifications.
- **Fairness:** Ensure that systems operate without bias, and treat all users equitably.
 - **Ethical and legal compliance:** Adhere to ethical principles and relevant legal standards.

Note that this is not an exhaustive list and that the relevance of each property depends on the specific context.

Trust and trustworthiness often rely on assumptions about adherence to best practices and regulatory compliance. TrustOps addresses this challenge through a continuous evidence collection methodology, leveraging trust-enabling tools to establish a robust trust model.¹⁰ For instance, TrustLLM evaluates LLMs across critical dimensions such as truthfulness, safety, and fairness, revealing performance variations and underscoring the complexities of balancing different properties.¹¹ Similarly, initiatives like Confiance.ai emphasize trustworthiness in high-stakes domains such as health care and defense, identifying key performance indicators to address critical needs.¹² To assist developers in enhancing code security early in the development process, trustworthiness models have been proposed that leverage security evidence, including software metrics and code smells, to identify vulnerable code units.¹³ These are just examples of frameworks and methodologies toward a strategic vision for software engineering that emphasizes trustworthiness and LLM-driven solutions.

LLMs IN TRUSTWORTHY SOFTWARE ENGINEERING

The advent of LLMs is initiating a paradigm change in software engineering by enabling advanced solutions for tasks such as code generation,

fault and vulnerability detection, and compliance assurance. For example, AutoSafeCoder integrates dynamic and static testing during LLM-based code generation, reducing vulnerabilities without sacrificing functionality.⁵ CodeCoR enhances code quality through a self-reflective multiagent system that refines LLM-generated code.¹⁴ SecRepair uses reinforcement learning to improve secure code generation and vulnerability detection.¹⁵ ChatUniTest, an automated testing framework, outperforms traditional tools like TestSpark in test coverage and accuracy through an adaptive focal context mechanism.¹⁶

Although existing works have explored the use of LLMs in software engineering,¹⁷ limited attention has been given to their full integration into the SDLC—from requirements gathering to deployment and post-deployment monitoring. This integration requires solutions that prioritize trustworthiness as a foundational element from the outset. Figure 1 exemplifies the role we foresee for LLMs in engineering trustworthy software.

Design

In the *requirements elicitation* phase, LLMs can facilitate a comprehensive and systematic approach to understand both functional and nonfunctional needs. Requirements elicitation traditionally relies on a combination of stakeholder interviews, document review, and use-case development.¹⁸ This process is time intensive and susceptible to mistakes, particularly when it comes to nonfunctional requirements. LLMs can enhance this process by automating the analysis of diverse sources, including stakeholder meeting transcriptions, user stories, and regulatory documents.

APPLYING AI TO SOFTWARE DEVELOPMENT



FIGURE 1. The role of LLMs in engineering trustworthy software. (Graphic(s) created using AI generation. For image credits, please see the “Acknowledgment” section of this article.)

Beyond elicitation, LLMs can also contribute to other aspects, such as *requirements prioritization and verification*. By analyzing dependencies, risk factors, and stakeholder preferences, LLM-based tools can help prioritize requirements, ensuring that critical system features are addressed early in the development lifecycle. Similarly, LLMs can assist in verifying

requirements by cross-checking them against domain-specific standards, historical project data, and pre-defined quality attributes, reducing ambiguities and ensuring alignment with project goals.

Automation can help in the identification of the relevant trustworthiness requirements, embedding relevant properties from the early stages of the

development process. For instance, in applications handling sensitive personal data, LLMs can extract privacy requirements directly from regulatory frameworks like the General Data Protection Regulation (GDPR), identifying needs for data minimization, encryption, and user consent. Similarly, LLMs can highlight security concerns by analyzing threat models. All this

will contribute to a foundational stage where the principles of a trustworthy system are established.

Once requirements are clearly defined, the focus shifts to *architectural design*, where LLMs can play a key role in guiding the creation of designs that consider and fulfill the relevant trustworthiness requirements. While architecture design typically emphasizes modularity, scalability, and maintainability, other aspects, such as resilience, fairness, data privacy, and security, are also essential to building trustworthy systems.¹⁹ While this is presently only a hypothesis, LLMs may be leveraged to analyze the requirements and recommend design patterns that align with the needs. For instance, in a distributed health-care application where data privacy is critical, an LLM might suggest a microservices-based architecture with isolated data processing modules, reducing data exposure and improving fault tolerance.

LLMs may also be potentially leveraged to understand the tradeoffs that arise between architectural choices, such as the balance between scalability and security, enabling architects to make informed decisions. Hypothetically, by embedding LLM-informed design principles, engineers would be able to establish an architectural foundation based on trustworthiness attributes, thus minimizing the need for retroactive modifications to address security and reliability concerns.

Development

During development, LLMs can significantly improve the process of writing code that adheres to best practices and trustworthiness principles.^{3,5} The role of LLMs in *code generation* should extend beyond mere syntax completion; these models should be

explored to generate code that incorporates security and reliability features, ensuring that developers do not inadvertently introduce bugs and vulnerabilities. For example, LLMs assisting in the development of a web application may include input validation mechanisms to prevent common vulnerabilities such as Structured Query Language (SQL) injection while also embedding proper error handling to ensure robustness. Furthermore, LLMs have the potential to provide suggestions for integrating secure and reliable libraries or third-party application programming interface (APIs), which can help reduce development time and ensure that trusted components are consistently used.

The role of LLMs becomes even more pronounced in *code analysis*, where exploring their capabilities for detecting flaws, vulnerabilities, and performance issues allows proactive quality assurance.⁵ LLMs can perform real-time code analysis by evaluating potential weaknesses as code is developed, thus identifying areas that may require rework. For instance, an LLM might detect an unprotected user input field and prompt the developer to implement validation checks. This prevents security and reliability issues from propagating through the code base.

In addition to code analysis, LLMs can support *automated testing* by generating test cases that evaluate both functional and nonfunctional aspects.^{5,16} By generating comprehensive test cases, LLMs can help ensure that code is rigorously evaluated against potential edge cases, reducing the likelihood of unforeseen errors. For example, in a financial application, LLMs might generate test cases that simulate high transaction volumes to verify the system's capacity to maintain

performance. Integrating LLM-based code analysis and automated testing into CI/CD will enable the continuous assessment of quality, ensuring that only code meeting trustworthiness standards is deployed.

Fault injection and vulnerability assessment are advanced alternatives for validating error detection and intrusion tolerance mechanisms.²⁰ LLMs can facilitate these processes by injecting faults or simulating vulnerabilities in the code, allowing engineers to observe its behavior under stress. This is particularly valuable in critical applications, where even minor faults can lead to catastrophic consequences. For example, LLMs can help create complex attack vectors to test the system's defensive mechanisms. LLMs can also be explored to define advanced fault and vulnerability models based on historical incident data from the same or other similar systems.

As code bases evolve over time, LLMs have the potential to support *refactoring* to maintain alignment with trustworthiness standards.²¹ For example, by detecting code patterns that may become security liabilities or performance bottlenecks, LLMs may prompt developers to revise outdated practices, keeping the code base reliable, secure, and easy to maintain. This contributes to the long-term quality and resilience of the system, promoting trustworthiness throughout the lifecycle.

In the context of *program repair*, LLMs have the potential to automatically detect and resolve defects, enhancing the trustworthiness of code bases.^{16,22} When issues arise, whether due to bugs, security vulnerabilities, or performance issues, LLMs can assist in generating context-aware fixes that align with best practices. For instance,

if a null pointer exception or an unhandled edge case is detected, LLMs can be used to generate corrective code. In scenarios where a bug might be security related, such as a missing access control check or inadequate input validation, LLMs can suggest security-enhancing code modifications to prevent exploitation. Program repair facilitated by LLMs may go beyond high-level fixes by understanding the broader implications of changes in an attempt to avoid introducing new vulnerabilities or compromising other parts of the code base.

Programming language migration is often necessary to modernize legacy systems, enhance performance, improve maintainability, or adopt more secure solutions. LLMs can play a role by facilitating the process of translating code from one language to another.²³ Migrating code manually is error prone, especially in complex systems where syntax, libraries, and underlying language semantics vary significantly. LLMs, trained on diverse language pairs and programming paradigms, can help automate this process by accurately converting language-specific constructs, functions, and data structures to their equivalents in the target language. For example, migrating a system from C/C++, which often relies on pointers and manual memory management, to Rust involves adapting to a strict memory management and ownership model that emphasizes safety and concurrency. LLMs have a great potential to automate this transition.

Deployment

IaC plays an essential role in ensuring consistent, reliable, and secure deployments. LLMs can help generate, validate, and troubleshoot deployment

configurations, reducing human error and expediting the deployment process.²⁴ For example, LLMs might generate IaC scripts that enforce access controls, apply network segmentation, or integrate encryption protocols, aligning infrastructure configurations with security and compliance standards. This capability extends across platforms as LLMs can translate configurations to fit multicloud or hybrid environments, making IaC more adaptable to varying deployment contexts.

Once the system is deployed, continuous monitoring and anomaly detection become key to maintaining trustworthiness. LLMs can contribute to the analysis of runtime data, such as system logs and user behavior, to identify deviations from expected patterns.²⁵ For instance, an LLM-based tool could flag unusual login patterns as a sign of unauthorized access. Through continuous monitoring, LLMs may not only support security but also enhance reliability and performance by tracking indicators such as memory usage, CPU load, and response times. This is particularly valuable in applications that require high availability, such as online transaction processing.

Effective issue management is vital for addressing incidents promptly and maintaining trustworthiness in the face of unexpected problems. LLMs will streamline issue management by automating the triage process, categorizing bugs, vulnerabilities, and incidents according to their impact and urgency.^{6,26} For example, an LLM might recognize a vulnerability affecting user authentication as critical, prompting the development team to prioritize it for immediate remediation. Automating prioritization will enable teams to allocate resources

efficiently and ensure that critical issues are addressed first.

LLMs can also assist in root cause analysis and fault localization by detecting patterns in system logs or failure reports that reveal the underlying issues.²⁷ In a distributed system, for instance, an LLM could analyze logs from multiple nodes to identify specific components or locations where network latency issues are impacting performance. This precise fault localization helps developers pinpoint problematic components. Additionally, LLMs can be leveraged to offer targeted remediation suggestions, providing actionable steps for developers to resolve issues quickly. This not only accelerates response times but also reduces the impact of incidents and fosters trustworthiness.

Assessment

Trustworthiness, as a dynamic and multifaceted quality, requires ongoing assessment to ensure that the system adapts to changing threats, user expectations, and regulatory requirements.⁹ In the context of software engineering, maintaining trustworthiness means that systems must not only meet initial standards but also evolve over time while simultaneously providing trustworthiness evidence.

A hypothetical application of LLMs is continuous trustworthiness assessment to enable proactive evaluation against the relevant metrics both during design and at runtime. During design, LLMs could play a crucial role in trustworthiness assessment by analyzing architectural choices and system specifications. Evaluating potential risks, such as vulnerabilities and scalability issues, and checking compliance with regulatory requirements can

provide insights that support informed decision making. For instance, LLMs might highlight areas where additional security controls are necessary or recommend architectural adjustments to enhance resilience. At runtime, LLM-based solutions may be leveraged to support trustworthiness by monitoring key events and metrics, such as security incidents or unusual user activity.

In the same context, LLMs may also have a role in the real-time calculation of trustworthiness scores and in deploying relevant dashboards that provide stakeholders with up-to-date information about the system's trustworthiness. Such scores are valuable for stakeholders who need assurance that the system consistently meets required standards. Continuously evaluating trustworthiness would enable a proactive approach that informs decision makers and supports corrective actions if trustworthiness scores indicate potential weaknesses. In practice, this continuous assessment framework would provide a foundation for long-term maintenance, enabling trustworthiness to evolve in response to internal changes and external conditions.

OPEN CHALLENGES

Integrating LLMs into software engineering holds great potential, but it also presents challenges that must be addressed to enable trustworthiness. In the following sections, we discuss several key challenges (see also [Figure 2](#)): integration of LLMs with established practices, accuracy and reliability, bias mitigation, explainability and interpretability, scalability and integration, standards and regulations, real-time adaptability, and ethics and privacy. To improve

readability, each section is structured in two paragraphs: the first highlighting the challenges and the second outlining potential directions for research.

LLMs and established practices

Integrating LLMs with existing software engineering practices, techniques,

existing techniques in a holistic manner. This includes developing hybrid approaches that combine LLM probabilistic reasoning with deterministic outputs and fine-tuned models tailored to tools like static analysis, penetration testing, and traceability solutions. Enhanced cross-tool interoperability and mechanisms for synchronization

[RESEARCH SHOULD FOCUS ON DESIGNING FRAMEWORKS THAT INTEGRATE LLMs WITH EXISTING TECHNIQUES IN A HOLISTIC MANNER.]

and tools is a mandatory step to achieve automation, improve efficiency, and address complexity challenges. However, this integration is not simple. For example, LLM-generated outputs may conflict with deterministic methods, creating inconsistencies. Also, the reliance of LLMs on contextual data can hinder the combination of structured and systematic outputs required by traditional software engineering tools. Taking static code analysis for vulnerability detection as an example, traditional static analysis tools rely on precise rule-based methods to identify specific patterns or issues in the code, producing deterministic results. If LLMs are introduced to improve this process, such as to generate explanations for detected vulnerabilities or to suggest fixes, they may produce probabilistic outputs that conflict with the findings of the static analysis tool.

To overcome these challenges, research should focus on designing frameworks that integrate LLMs with

between LLMs and traditional tools also need to be developed.

Accuracy and reliability

Accuracy is foundational for LLMs in trustworthy software engineering. As LLMs are deployed across the different lifecycle stages, it becomes of critical importance to ensure that outputs are precise and free from errors. Unlike conventional automation tools, LLMs generate probabilistic outputs based on patterns learned from large datasets, which can lead to both contextually correct and misleading suggestions.¹¹ For instance, during code generation, LLMs may provide solutions that appear syntactically correct but introduce security flaws or performance bottlenecks due to insufficient contextual understanding.³

Research is needed to develop mechanisms that validate and verify LLM outputs, especially in business- and mission-critical applications where even minor issues can lead to significant

APPLYING AI TO SOFTWARE DEVELOPMENT

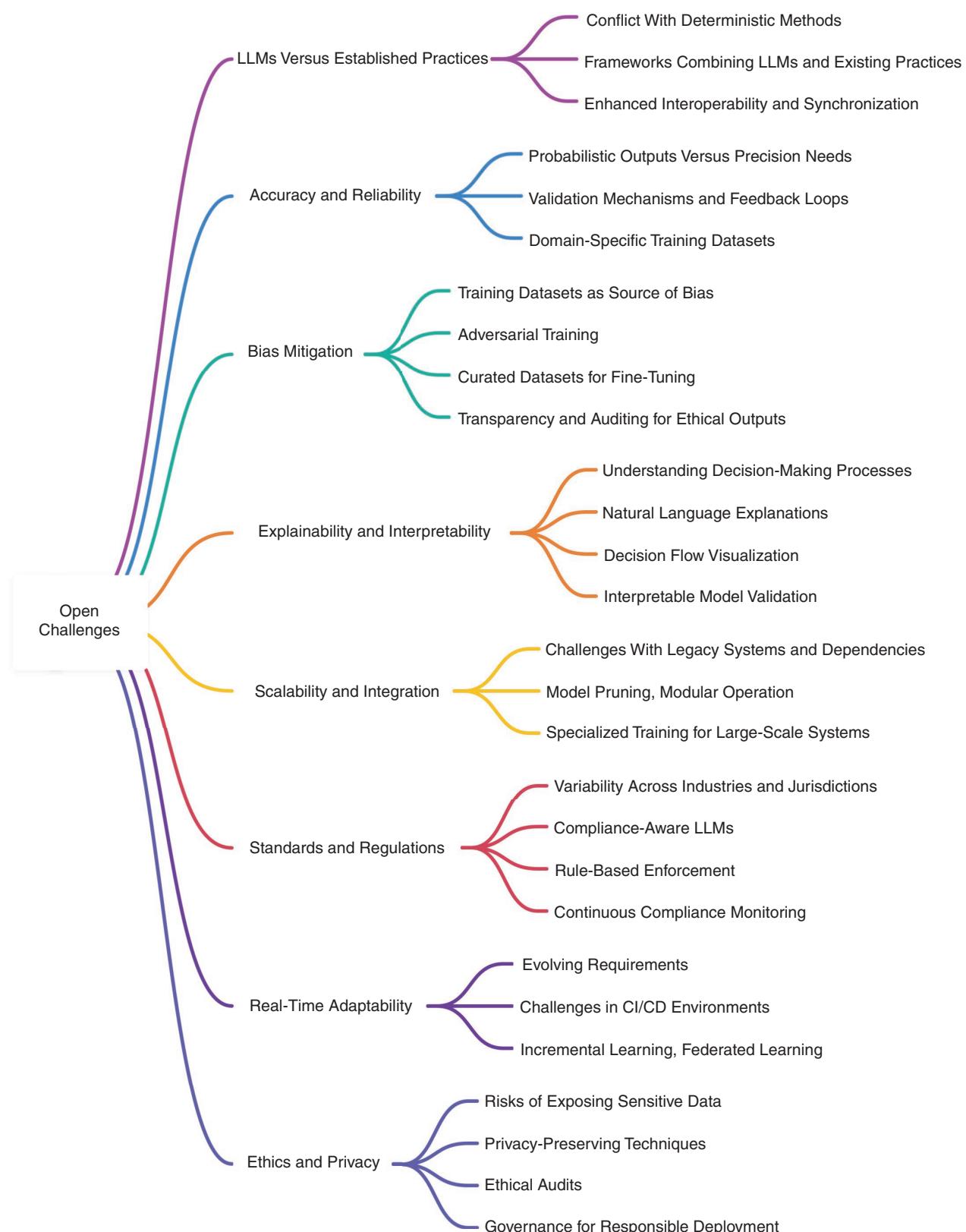


FIGURE 2. Challenges for LLMs in engineering trustworthy software. (Graphic(s) created using AI generation. For image credits, please see the "Acknowledgment" section of this article.)

consequences. Techniques such as output validation layers, specialized training datasets tailored for domain-specific contexts, and hybrid systems combining rule-based checks with LLM-driven suggestions could enhance accuracy. Additionally, integrating feedback loops, where human experts review and correct LLM outputs, may provide continuous refinement to improve accuracy and reliability.

Bias mitigation

Bias in LLMs is a major challenge as these models are trained on datasets from various sources, some containing biases related to language, coding practices, or even ethical aspects. For example, LLMs trained on open source code might inherit biases from less secure or less efficient coding practices commonly found in certain community-contributed libraries. This can result in LLMs suggesting outdated or nonstandard approaches that undermine trustworthiness.

Although various solutions have been proposed to reduce specific biases, such as biases in internal components of LLMs and biases arising from user interactions,¹¹ their specific application to trustworthy software engineering requires further study. To address these challenges, research must focus on developing effective detection techniques to analyze LLM outputs for deviations from established best practices or unintended patterns. Techniques such as adversarial training, which trains LLMs to detect and counteract their own biases, and fine-tuning models with curated bias-free datasets should be explored to minimize these risks. Moreover, ensuring transparency in training data sources and conducting audits of LLM-generated outputs are

essential steps to identify and address patterns of bias.¹¹

Explainability and interpretability

The opaque nature of LLM decision-making processes is a significant obstacle to building trust. Developers and stakeholders need to understand the rationale behind LLM-generated outputs, especially in critical domains like health care, finance, and cybersecurity. However, the closed-box nature of LLMs complicates interpretability, making it challenging to understand why specific architectural decisions, code patterns, or testing suggestions are recommended. For instance, if an LLM recommends a specific software requirement, a corresponding explanation should indicate how this choice aligns with the system objectives.

Research into explainable AI methods specifically tailored for LLMs is essential to improve transparency and interpretability. Explainability can involve generating explanations alongside technical recommendations that trace how an LLM arrived at a particular suggestion.¹¹ Additionally, enhancing transparency in the internal mechanisms and decision-making processes of LLMs can be achieved through approaches like the Chain of Thought paradigm.²⁸ This approach lays out the intermediate steps and reasoning used by the model to reach its conclusions, making its operations significantly more interpretable for human users. Additionally, explainable AI should offer frameworks and tools that help explain the internal mechanisms, knowledge storage, and decision-making pathways of LLMs, promoting a deeper understanding and fostering greater trust in these models.¹¹

Scalability and integration

LLMs will face challenges when applied to large-scale systems,¹⁷ especially in complex ecosystems that rely on numerous third-party components. Modern software systems often include legacy code bases, complex interdependencies, and integration with cloud platforms, microservices, and external APIs. Embedding LLMs into these environments requires advanced research into scalability and interoperability.

Ensuring that LLMs can handle the volume and complexity of information regarding large-scale systems is crucial for reliable integration. For example, LLMs will struggle with inconsistencies or outdated dependencies within legacy systems, resulting in erroneous recommendations or failure to adhere to best practices. Techniques like model pruning, where nonessential parameters are removed to optimize performance, or partitioning large models to operate independently across different software modules could help address scalability issues. Additionally, specialized LLMs trained on legacy code patterns may support the ongoing maintenance of older code bases.

Standards and regulations

As software systems increasingly interact with sensitive data and critical infrastructures, compliance with standards and regulations is essential for trustworthiness. However, regulatory compliance often varies across jurisdictions and industries, from the GDPR in Europe to the Health Insurance Portability and Accountability Act (HIPAA) in health care and the Payment Card Industry Data Security Standard (PCI DSS) in finance. LLMs must be able to navigate this complex

ABOUT THE AUTHOR

MARCO VIEIRA is a professor at the University of North Carolina at Charlotte, Charlotte, NC 28223 USA. His research interests include dependability and security assessment, fault injection, and software testing. Vieira received a Ph.D. in informatics engineering from the University of Coimbra, Portugal. He is the chair of the IFIP WG 10.4 on Dependable Computing and Fault Tolerance. He is a Member of IEEE. Contact him at marco.vieira@charlotte.edu.

landscape and generate code, recommendations, and configurations that align with the relevant regulations. Concerns have already been raised regarding their compliance with existing data privacy legislation, such as the GDPR,²⁹ highlighting the need for LLMs to effectively address these challenges to uphold trust and reliability.

Research is needed to develop compliance-aware LLMs that can recognize regulatory requirements and integrate them into their outputs. For instance, an LLM-generating IaC configuration should enforce data encryption or role-based access control based on the applicable regulations. Rule-based techniques should be researched to enforce regulatory constraints to be directly applied to LLM outputs. Furthermore, fine-tuning models with compliance-specific datasets could ensure that LLM-driven software engineering adheres to standards. Compliance auditing tools should also be developed to continuously monitor adherence to regulations.

Real-time adaptability

CI/CD calls for adaptability in LLM-driven software engineering processes, especially as requirements evolve throughout the lifecycle. As

software is deployed at rapid intervals, LLM-based solutions need to promptly provide relevant and accurate feedback. However, the latency of large models, combined with the risk of outdated recommendations as both software and requirements evolve, presents significant challenges. Also, adapting to evolving requirements is mandatory to ensure that the outputs remain aligned with existing business goals and constraints.

Research should explore LLM architectures or modular model deployment strategies that can deliver near-instant feedback. Techniques like incremental learning, where LLMs are updated continuously with recent data, can help ensure that recommendations remain relevant. Additionally, exploring federated learning, where models are trained and deployed across decentralized locations, may help LLMs keep pace with the rapid cycles of CI/CD pipelines.

Ethics and privacy

The ethical implications of using LLMs in software engineering, particularly regarding privacy and user data, require careful consideration. Models trained on broad datasets may inadvertently expose sensitive information

or generate outputs that compromise user privacy, especially if data sanitization is inadequate.^{11,29} Furthermore, ensuring that LLMs operate ethically is extremely important in sectors that involve sensitive applications, such as health care or law enforcement.

Researching the applicability of privacy-preserving techniques to ensure that LLMs generate outputs without revealing sensitive data is mandatory. Additionally, regular ethical audits and compliance with privacy guidelines can further support the responsible deployment of LLMs. Transparent governance and ethical controls can help align LLM practices with privacy and data security values, fostering trust in LLM-driven software solutions.

The potential for LLMs to transform software engineering is enormous but requires advanced research to tackle challenges like accuracy, bias mitigation, explainability, and scalability. At the same time, there is a critical need for practical studies that test LLM-driven approaches in real-world software development environments. These studies should explore the scalability and accuracy of LLMs under diverse conditions and in large-scale systems, providing insights into their practical applicability.

Research should focus on improving LLM explainability to build user trust, mitigating biases, and designing compliance-aware models aligned with regulations. Practitioners should adopt hybrid approaches combining probabilistic LLM outputs with deterministic tools for consistency. Future efforts should explore integrating LLMs with legacy systems and existing tools to expand their applicability. Strategies for enhancing real-time

adaptability while maintaining regulatory compliance are needed for deploying LLMs effectively in high-stakes domains, enabling their reliable use in diverse and evolving software development environments.

While the transformative potential of LLMs in software engineering is evident, their integration must also be guided by considerations of environmental sustainability. The energy consumption associated with training and deploying LLMs poses a significant challenge, contributing to a larger carbon footprint. Future research and development should prioritize energy-efficient methods and sustainable practices to ensure that the environmental costs of LLMs remain justified by their benefits, enabling their responsible adoption in the software engineering lifecycle. □

ACKNOWLEDGMENT

The author acknowledges that ChatGPT 4o was used to improve the writing and for extracting from the text a draft version of the mind maps presented in Figures 1 and 2.

REFERENCES

1. T. B. Brown et al., "Language models are few-shot learners," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, Red Hook, NY, USA, 2020, pp. 1877–1901.
2. J.-H. Cho, S. Xu, P. M. Hurley, M. Mackay, T. Benjamin, and M. Beaumont, "STRAM: Measuring the trustworthiness of computer-based systems," *ACM Comput. Surv.*, vol. 51, no. 6, pp. 1–47, Nov. 2019, doi: [10.1145/3277666](https://doi.org/10.1145/3277666).
3. L. Chen et al., "A survey on evaluating large language models in code generation tasks," 2024, *arXiv:2408.16498*.
4. R. Dhar, K. Vaidhyanathan, and V. Varma, "Can LLMs generate architectural design decisions? - An exploratory empirical study," in *Proc. IEEE 21st Int. Conf. Softw. Archit. (ICSA)*, Hyderabad, India, 2024, pp. 79–89, doi: [10.1109/ICSA59870.2024.00016](https://doi.org/10.1109/ICSA59870.2024.00016).
5. A. Nunez, N. T. Islam, S. Jha, and P. Najafirad, "AutoSafeCoder: A multi-agent framework for securing LLM code generation through static analysis and fuzz testing," 2024, *arXiv:2409.10737*.
6. W. Tao, Y. Zhou, Y. Wang, W. Zhang, H. Zhang, and Y. Cheng, "MAGIS: LLM-based multi-agent framework for GitHub issue resolution," 2024, *arXiv:2403.17927*.
7. J. Slemrod and P. Katuscak, "Do trust and trustworthiness pay off?" *Nat. Bureau Econ. Res.*, Cambridge, MA, USA, Working Paper 9200, 2002. [Online]. Available: <https://www.bus.umich.edu/otpr/WP2002-8%20do%20trust%20and%20trustworthiness.pdf>
8. M. Nami and W. Suryn, "Software trustworthiness: Past, present and future," in *Trustworthy Computing and Services*, Y. Yuan, X. Wu, and Y. Lu, Eds. Berlin, Germany: Springer-Verlag, 2013, pp. 1–12.
9. N. Medeiros, N. R. Ivaki, P. N. D. Costa, and M. P. A. Vieira, "Towards an approach for trustworthiness assessment of software as a service," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, Honolulu, HI, USA, 2017, pp. 220–223, doi: [10.1109/IEEE.EDGE.2017.39](https://doi.org/10.1109/IEEE.EDGE.2017.39).
10. E. Brito, F. Castillo, P. Pullonen-Raudvere, and S. Werner, "TrustOps: Continuously building trustworthy software," 2024, *arXiv:2412.03201*.
11. Y. Huang et al., "Position: TrustLLM: Trustworthiness in large language models," in *Proc. 41st Int. Conf. Mach. Learn.*, 2024, pp. 20,166–20,270.
12. J. Mattioli et al., "An overview of key trustworthiness attributes and KPIs for trusted ML-based systems engineering," *AI Ethics*, vol. 4, no. 1, pp. 15–25, Feb. 2024, doi: [10.1007/s43681-023-00394-2](https://doi.org/10.1007/s43681-023-00394-2).
13. N. Medeiros, N. Ivaki, P. Costa, and M. Vieira, "Trustworthiness models to categorize and prioritize code for security improvement," *J. Syst. Softw.*, vol. 198, Apr. 2023, Art. no. 111621, doi: [10.1016/j.jss.2023.111621](https://doi.org/10.1016/j.jss.2023.111621).
14. R. Pan, H. Zhang, and C. Liu, "CodeCoR: An LLM-based self-reflective multi-agent framework for code generation," 2025, *arXiv:2501.07811*.
15. N. T. Islam et al., "LLM-powered code vulnerability repair with reinforcement learning and semantic reward," 2024, *arXiv:2401.03374*.
16. Y. Chen, Z. Hu, C. Zhi, J. Han, S. Deng, and J. Yin, "ChatUniTest: A framework for LLM-based test generation," in *Companion Proc. 32nd ACM Int. Conf. Found. Softw. Eng. (FSE)*, Porto de Galinhas, Brazil, 2024, pp. 572–576, doi: [10.1145/3663529.3663801](https://doi.org/10.1145/3663529.3663801).
17. X. Hou et al., "Large language models for software engineering: A systematic literature review," *ACM Trans. Softw. Eng. Method.*, vol. 33, no. 8, pp. 1–79, Dec. 2024, doi: [10.1145/3695988](https://doi.org/10.1145/3695988).
18. P. A. Laplante and M. Kassab, *Requirements Engineering for Software and Systems*, 4th ed. Boca Raton, FL, USA: Auerbach, 2022.
19. D. Garlan and M. Shaw, "An introduction to software architecture," in *Advances in Software Engineering and Knowledge Engineering*, V. Ambriola and G. Tortora Ed. Singapore: World Scientific, 1993, pp. 1–39.
20. R. Natella, D. Cotroneo, and H. S. Madeira, "Assessing dependency with software fault injection: A

- survey," *ACM Comput. Surv.*, vol. 48, no. 3, pp. 1–55, Feb. 2016, doi: [10.1145/2841425](https://doi.org/10.1145/2841425).
21. A. Shirafuji, Y. Oda, J. Suzuki, M. Morishita, and Y. Watanobe, "Refactoring programs using large language models with few-shot examples," in *Proc. 30th Asia-Pacific Softw. Eng. Conf. (APSEC)*, Seoul, South Korea, 2023, pp. 151–160, doi: [10.1109/APSEC60848.2023.00025](https://doi.org/10.1109/APSEC60848.2023.00025).
22. M. Jin et al., "InferFix: End-to-end program repair with LLMs," in *Proc. 31st ACM Joint Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng. (ESEC/FSE)*, San Francisco, CA, USA, 2023, pp. 1646–1656, doi: [10.1145/3611643.3613892](https://doi.org/10.1145/3611643.3613892).
23. J. Hong and S. Ryu, "Type-migrating C-to-Rust translation using a large language model," *Empirical Softw. Eng.*, vol. 30, no. 1, Oct. 2024, Art. no. 3, doi: [10.1007/s10664-024-10573-2](https://doi.org/10.1007/s10664-024-10573-2).
24. J. Diaz-De-Arcaya, J. López-De-Armentia, G. Zárate, and A. I. Torre-Bastida, "Towards the self-healing of infrastructure as code projects using constrained LLM technologies," in *Proc. 5th ACM/IEEE Int. Workshop Autom. Program. Repair (APR)*, 2024, pp. 22–25, doi: [10.1145/3643788.3648014](https://doi.org/10.1145/3643788.3648014).
25. Z. Yu et al., "MonitorAssistant: Simplifying cloud service monitoring via large language models," in *Companion Proc. 32nd ACM Int. Conf. Found. Softw. Eng. (FSE)*, 2024, pp. 38–49, doi: [10.1145/3663529.3663826](https://doi.org/10.1145/3663529.3663826).
26. X. Du, Z. Liu, C. Li, X. Ma, Y. Li, and X. Wang, "LLM-BRC: A large language model-based bug report classification framework," *Softw. Qual. J.*, vol. 32, no. 3, Sep. 2024, pp. 985–1005, doi: [10.1007/s11219-024-09675-3](https://doi.org/10.1007/s11219-024-09675-3).
27. S. Kang, G. An, and S. Yoo, "A quantitative and qualitative evaluation of LLM-based explainable fault localization," in *Proc. ACM Softw. Eng.*, Jul. 2024, vol. 1, no. FSE, pp. 1424–1446, doi: [10.1145/3660771](https://doi.org/10.1145/3660771).
28. J. Wei et al., "Chain-of-thought prompting elicits reasoning in large language models," in *Proc. 36th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, New Orleans, LA, USA, 2022, pp. 24,824–24,837.
29. Z. Sun, "A short survey of viewing large language models in legal aspect," 2023, [arXiv:2303.09136](https://arxiv.org/abs/2303.09136).



IEEE Computer Graphics and Applications bridges the theory and practice of computer graphics. Subscribe to CG&A and

- stay current on the latest tools and applications and gain invaluable practical and research knowledge,
- discover cutting-edge applications and learn more about the latest techniques, and
- benefit from CG&A's active and connected editorial board.

Digital Object Identifier 10.1109/MC.2025.3573133





©SHUTTERSTOCK.COM/VARUNWUU

The Return of the Magic Triangle

Christof Ebert , Vector Consulting Services

The magic triangle is back. Innovation and quality are at stake with cost being a major focus. Learn from industry leaders how to navigate this challenge without risking your future competitiveness.

The magic triangle is back. It has been used for decades to illustrate the goal-conflicts that mostly undermine the success of an endeavor^{1,2}

- › Time to finish,
- › Cost and effort,
- › Quality and scope.

These three conflicting goals are symbolized by the corners of a triangle. Changing one, such as scope or deadline, will immediately impact the two others. Balancing these three goals in software and IT projects is seen as challenging, thus the word "magic." With more structured industrialized software engineering and management methods, fueled by capability maturity model and agile development,

it became clear that quality and time do not necessarily impact cost negatively, considering that technical debt and rework is the consequence of overly fast development.

At year-end 2024, Vector Consulting surveyed industry experts about their challenges and resolutions. Not surprisingly, competitive pressure dominates the feedback. We observe a new magic triangle of cost, quality, and innovation. Four risks were mentioned, ranging from misunderstood agility to the focus on "ticket-driven software" and "everything as code" with insufficient test and hardening. Rework, technical debt, and liability risks increase and will impact cost.

The answers reflect the perceived competitive pressure worldwide. Figure 1 shows the perceived challenges across industries and worldwide. The horizontal axis shows short-term challenges, while the vertical axis represents medium-term obstacles. Each respondent was able to name up to five challenges in both dimensions. The total is therefore more than 100%. We observe a magic triangle of cost, quality, and innovation.

In our interviews with software and IT developers and managers four risks were mentioned that result from this magic triangle:

FROM THE EDITOR

Software and IT companies are trapped in a new magic quadrant. Balancing innovation, cost, and quality is key to software success. With cost being a major pressure from markets, technical debt is growing. Our article with results from an industry survey shows how to mitigate the magic triangle without risk of sacrificing your future competitiveness. Learn from industry leaders and benefit from the transfer suggestions. A video podcast for this article is available at <https://youtu.be/D-3GmNCJ4Dc>. —Christof Ebert

- **Innovation trap:** Tech companies are struggling with the high demands for steady innovation with a simultaneous lack of capacity and expertise. Impact: Downward spiral for established companies that do not bring enough new products to market.
- **Cost trap:** Ad-hoc cost reduction and a misunderstood “right-sourcing” might improve short-term cash flow but with higher project and product risks. Impact: Insufficient focus on quality causes rework and callback with additional expenses and delays.
- **Agility trap:** Perceived pragmatic patchwork processes invite “anything goes” as processes are not taken seriously.

Impact: Increasing governance risks with high risk of supplier audits, competitive disadvantages, and recalls and associated costs.

- **IT trap:** Practices such as everything as code and ticket-based development are used in complex systems. They accelerate development, but at the cost of insufficient contextual understanding. Impact: Growing technical debt and security exposure.

The current one-dimensional focus on only cost reduction is leading to a downward spiral, as our interview partners pointed out. Engineers and companies know in theory what they should do in practice, but they

often lack the necessary processes and management support to put it into practice. The mere focus on cost reduction alone undermines the ability of companies to innovate. Given the parallel loss of competences, our interview partners see quality decreasing and technical debt increasing. Misunderstood agility and diminishing process capabilities contribute to commercial risks.¹

Take cybersecurity, for example: IT solutions from the cookbook are of little help in critical environments. What is needed here are concrete instructions on how the standards can be implemented efficiently and in interaction with each other. Cybersecurity and safety can only be achieved sustainably across variants and versions if systems engineering and lifecycle management are linked.

Interview partners emphasize that systematic development of critical systems is a top priority with growing safety hazards and cybersecurity threats. At the same time, engineering capacity is reduced with tasks remaining unchanged. To mitigate Generative AI (GenAI) “copilots” are used as development accelerators but create risks on quality, maintainability, and intellectual property. Examples were brought forward to us, indicating that in search for more efficiency some traditional software tasks are currently enhanced by GenAI. Proper development of critical systems is of increasing relevance as it needs professional process and deep technology expertise, which a chatbot will not have, and which from liability and insurance perspectives cannot be outsourced to such chatbot.

MITIGATING THE RISKS OF THE MAGIC TRIANGLE

We have asked our survey participants what they recommend as takeaways to improve the risks of the new magic triangle. Here are some hands-on guidance and key takeaways from our consulting projects (see also Guide to Grow). Use them for transfer to your own environments:



FIGURE 1. The magic triangle of competitiveness: innovation, cost, and quality.

Artificial intelligence (AI) everywhere: AI is increasingly integrated in workflows and products across industries. Digital assistants or copilots enhance productivity and user experience. Current AI models demonstrate improved contextual awareness, better natural language processing, and greater adaptability, enabling smarter, more intuitive interactions across various platforms and devices.

Sustainability: As environmental concerns grow, companies focus on sustainable product design, energy-efficient technologies, and ensuring longer device lifespans. Green IT plays a pivotal role, impacting all software domains. Sustainability will evolve especially in Europe and East Asia, a key competitive differentiator, shaping corporate strategies and influencing consumer purchasing decisions worldwide.

Standard efficiency: Governments worldwide are implementing evolving AI and data privacy regulations, creating compliance challenges for global businesses. Differences in regional laws, such as the EU's AI Act and various national cybersecurity mandates, require companies to navigate complex legal landscapes. In the Internet of Things sector, the absence of standardized protocols further complicates seamless device integration across ecosystems, slowing widespread adoption and increasing costs for developers and manufacturers.

Verification and validation: A major cost driver is verification and validation, which today eats over 50% of lifecycle cost. Companies must rework their testing processes and infrastructure. Cost efficiency of R&D and engineering is critical, which demands agile processes, and a much better connect of standards as is currently visibly. Distributed R&D must improve with global IT and SW supply chains ensuring functional safety and compliance of components.

Cybersecurity: The rapid adoption of AI raises significant cybersecurity concerns, particularly around

GUIDE TO GROW

TAKEAWAYS FROM INDUSTRY TRENDS

- » The magic triangle has returned. Balancing innovation, quality and cost is the key to staying competitive in software-driven industries. Today, with rising complexity, shrinking budgets, and global competition, companies must revisit these principles with modern software engineering methods.
- » Shift from cost-cutting to value-driven efficiency. Often, inexperienced software and IT managers mistakenly equate efficiency with cost-cutting, leading to technical debt, slower innovation, and poor quality. Utilize software factories, model-driven engineering, and automation to cut waste while enhancing productivity.
- » Avoid misunderstood agility. A major cause of inefficiencies is the drive toward fragmented so-called agile mini steps. Purely reactive agile workflows without proper architectural planning contribute to technical debt but not to sustainable competitiveness. Strong architecture, upfront design, and systematic requirements engineering lead to faster development cycles, fewer defects, and better long-term scalability.
- » Software factories: To stay ahead, companies must treat software as a strategic asset rather than an afterthought. Software factories—with continuous integration and deployment (CI/CD), AI-driven automation, and component reuse—reduce effort while boosting quality. This is especially critical in automotive, medical, and embedded systems, where software complexity is skyrocketing.
- » From R&D to ROI: Successful companies align their R&D investments with business value. Leaders in software-intensive industries achieve faster innovation cycles by connecting software engineering with business strategy. This demands strong product management and robust technology and IT strategies, together with focus on agile quality-first.

TRANSFER QUESTIONS

- » How does your current software development balance innovation, cost, and quality? Where do you see gaps that could be improved with efficient engineering practices?
- » Are you focusing on cost-cutting or true efficiency, such as automating processes, improving quality, reducing rework? What steps can you take to move from short-term savings to long-term competitiveness?
- » What elements of a "software factory", for example, automation, CI/CD, or modular architectures could help you scale innovation and improve software quality and innovation without overheads?
- » How well is your technology strategy aligned with business objectives? What changes would help ensure your software investments drive measurable competitive advantage?

misinformation, deepfake technology, and privacy breaches. AI-generated content can be exploited for deception, posing risks to your products, such as

introducing unwanted backdoors. As more devices become interconnected, the attack surface for cyberthreats expands, demanding robust AI-driven

security measures. Companies need to deploy real-time threat detection systems and enhanced encryption technologies to safeguard their IPR and sensitive data.

Process and methods: Many companies lack systematic methodology. Go

open sharing of thoughts, ideas, and concerns. This means to get back into the office and physically interact with colleagues. Remote work is ok for mechanical tasks and project management, but not for thinking out of the box and generating new ideas.

The mere focus on cost reduction alone undermines the ability of companies to innovate.

beyond technology competence. Engineers love the technology stack and certainly know what is relevant for their work. Yet they often forget to look beyond today into evolving business needs. A case in point is the growing impact of GenAI, both as productivity driver, for example, code generation and test support, and as a risk, for example, intellectual property rights and degrading own competences.

Competence growth: Agree on individual and team learning targets at least on an annual basis in the performance reviews. There is no evolution without measurable targets and actual feedback. Check the learning and commit to measurable learning objectives. Establish systematic competence evolution by competence grid for each basic role. To motivate growth a competency matrix should show an advancement path from junior to expert and how to grow with self-learning, mentoring, classes, hands-on projects, etc.

Knowledge management: Use innovative technologies for knowledge management. GenAI is today often considered as a major knowledge source, yet needs not only intelligent prompts, but also careful filtering to highlight what matters and withdraw hallucinations. While it is relevant to deploy standard tools, such as SharePoint and collaboration tools, personal and organizational processes to share and grow competencies are more relevant.

Learning culture: Encourage informal interactions across the team and

Ensure that formal meetings and communication are timely, transparent, and widespread. Many meetings have long topic lists but no takeaways and no sustainable learning. Start with what one or more persons have learned in the past week. Use open issue lists to systematically follow up and close items.

WHERE DO WE GO FROM HERE?

Our business environment is increasingly uncertain with rapid changes. To thrive in this time of continuous change, companies and each single engineer need resilience to accept sudden changes and agility to immediately exploit the emerging opportunities.³ This demands a continuous learning to adapt to change and to adopt new practices. Technology evolution is driven by AI, connectivity, and sustainability. Companies and policymakers must collaborate to ensure responsible innovation and bridge the technological divide. As industries navigate these challenges, they must seize opportunities to create a more inclusive and sustainable digital future.

The future belongs to companies that invest in efficient, high-quality, and strategically aligned software development. Cutting corners may provide short-term savings, but it destroys innovation and competitiveness in the long run. To mitigate the evolving industry and economic challenges demand more competence than most companies bring.

Pragmatism matters. Adele Goldberg, the famous computer scientist who currently celebrates her 80th birthday, remarked: "Don't ask whether you can do something, but how to do it." Together with Alan Kay of PARC, she was a thought leader toward what we today label object-orientation, design patterns, and user experience. In translating a strong vision into actual innovations and products, she is a role model in computer science and software engineering. Delivering actual products, she had a profound impact in reducing the risk of the ancient magic triangle. The major takeaway in current economic turmoil is to connect from needs to useful products—and not stop halfway. □

ACKNOWLEDGMENT

We want to thank all survey participants for supporting this study and thus ensuring validity.

REFERENCES

1. "Software engineering body of knowledge (SWEBOK)," IEEE Comput. Soc. Press, Los Alamitos, CA, USA, ISO Tech. Rep. 19759, 2024. [Online]. Available: <https://www.computer.org/education/bodies-of-knowledge/software-engineering>
2. C. Ebert and M. Smouts, "Tricks and traps of initiating a product line concept in existing products," in Proc. 25th Int. Conf. Softw. Eng. (ICSE), Portland, OR, USA, 2003, pp. 520–527, doi: [10.1109/ICSE.2003.1201231](https://doi.org/10.1109/ICSE.2003.1201231).
3. G. Doukidis, D. Spinellis, and C. Ebert, "Digital transformation," IEEE Softw., vol. 37, no. 5, pp. 13–21, Sep. 2020, doi: [10.1109/MS.2020.2999969](https://doi.org/10.1109/MS.2020.2999969).

CHRISTOF EBERT is the managing director of Vector Consulting Services, Stuttgart, 70499, Germany. He is a Senior Member of IEEE. Contact him at <https://www.linkedin.com/in/christofebert> or christof.ebert@vector.com.



PURPOSE: Engaging professionals from all areas of computing, the IEEE Computer Society sets the standard for education and engagement that fuels global technological advancement. Through conferences, publications, and programs, IEEE CS empowers, guides, and shapes the future of its members, and the greater industry, enabling new opportunities to better serve our world.

OMBUDSMAN: Contact ombudsman@computer.org.

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

PUBLICATIONS AND ACTIVITIES

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

Periodicals: The IEEE CS publishes 12 magazines, 18 journals

Conference Proceedings & Books: Conference Publishing Services publishes more than 275 titles every year.

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

Technical Communities: TCs provide professional interaction in more than 30 technical areas and directly influence computer engineering conferences and publications.

Conferences/Education: The IEEE CS holds more than 215 conferences each year and sponsors many educational activities, including computing science accreditation.

Certifications: The IEEE CS offers three software developer credentials.

AVAILABLE INFORMATION

To check membership status, report an address change, or obtain information, contact help@computer.org.

IEEE COMPUTER SOCIETY OFFICES

WASHINGTON, D.C.:

2001 L St., Ste. 700,
Washington, D.C. 20036-4928

Phone: +1 202 371 0101

Fax: +1 202 728 9614

Email: help@computer.org

LOS ALAMITOS:

10662 Los Vaqueros Cir.,
Los Alamitos, CA 90720

Phone: +1 714 821 8380

Email: help@computer.org

IEEE CS EXECUTIVE STAFF

Executive Director: Melissa Russell

Director, Governance & Associate Executive Director:
Anne Marie Kelly

Director, Conference Operations: Silvia Ceballos

Director, Information Technology & Services: Sumit Kacker

Director, Marketing & Sales: Michelle Tubb

Director, Membership Development: Eric Berkowitz

Director, Periodicals & Special Projects: Robin Baldwin

Digital Object Identifier 10.1109/MC.2025.3561966

IEEE CS EXECUTIVE COMMITTEE

President: Hironori Washizaki

President-Elect: Grace A. Lewis

Past President: Jyotika Athavale

Vice President: Nils Aschenbruck

Secretary: Yoshiko Yasuda

Treasurer: Darren Galpin

VP, Member & Geographic Activities: Andrew Seely

VP, Professional & Educational Activities: Cyril Onwubiko

VP, Publications: Charles (Chuck) Hansen

VP, Standards Activities: Edward AU

VP, Technical & Conference Activities: Terry Benzel

2025–2026 IEEE Division VIII Director: Cecilia Metra

2024–2025 IEEE Division V Director: Christina M. Schober

2025 IEEE Division V Director-Elect: Leila De Floriani

IEEE CS BOARD OF GOVERNORS

Term Expiring 2025:

İlkay Altıntaş, Mike Hinckey, Joaquim Jorge, Rick Kazman, Carolyn McGregor, Andrew Seely

Term Expiring 2026:

Megha Ben, Terry Benzel, Mrinal Karvir, Andreas Reinhardt, Deborah Silver, Yoshiko Yasuda

Term Expiring 2027:

Sven Dickinson, Alfredo Goldman, Daniel S. Katz, Yuhong Liu, Ladan Tahvildari, Damla Turgut

IEEE EXECUTIVE STAFF

Executive Director and COO: Sophia Muirhead

General Counsel and Chief Compliance Officer:
Ahsaki Benion

Chief Human Resources Officer: Cheri N. Collins Wideman

Managing Director, IEEE-USA: Russell Harrison

Chief Marketing Officer: Karen L. Hawkins

Managing Director, Publications: Steven Heffner

Staff Executive, Corporate Activities: Donna Hourican

Managing Director, Member and Geographic Activities:
Cecelia Jankowski

Chief of Staff to the Executive Director: TBA

Managing Director, Educational Activities: Jamie Moesch

IEEE Standards Association Managing Director: Alpesh Shah

Chief Financial Officer: Kelly Lorne

Chief Information Digital Officer: Jeff Strohschein

Managing Director, Conferences, Events, and Experiences:
Marie Hunter

Managing Director, Technical Activities: Mojdeh Bahar

IEEE OFFICERS

President & CEO: Kathleen A. Kramer

President-Elect: Mary Ellen Randall

Past President: Thomas M. Coughlin

Director & Secretary: Forrest D. Wright

Director & Treasurer: Gerardo Barbosa

Director & VP, Publication Services & Products: W. Clem Karl

Director & VP, Educational Activities: Timothy P. Kurzweg

Director & VP, Membership and Geographic Activities:
Antonio Luque

Director & President, Standards Association:

Gary R. Hoffman

Director & VP, Technical Activities: Dalma Novak

Director & President, IEEE-USA: Timothy T. Lee

Computer Education in the Machine Learning Era: Intelligent Systems

Brian M. Sadler^{ID}, University of Texas at Austin

In the machine learning (ML) Era computing will become synonymous with intelligence, blending classical and ML-based methods. This points to fundamental changes in computer education.

As profound as was the progression from analog to digital computing, so is the current progression into the era of machine learning (ML). The early ML Era is marked by two key trends, a technology progression to systems, and the use of data-driven compute architectures.

Together, these are enabling all kinds of intelligent systems R&D and evolving the foundations of computer education.

Stemming from the digital era, technology consolidation has progressed over the past few decades into smaller highly capable devices and (mobile) platforms, linked through worldwide communications with dramatically increasing bandwidth and corresponding lower latency. Much of computer science and engineering research was previously focused on component



technologies, feeding into system development typically carried out in industry. Today, beyond “system silos,” the combination and convergence of microscale component technologies is commonplace. Sensing, perception, action, and networking are readily combined in secondary school projects, and highly multidisciplinary complex systems are common in graduate research.

A networked consolidation trend also holds at macroscale. Vast on-demand computing is readily available and becoming much more so as compute center infrastructure continues to grow. Cellular communications trends include emerging standards for extremely low latency messaging, distributed and edge computing, and increasing communications capacity with rapidly maturing autonomous collaborative unmanned aerial vehicles and intelligent reflective surfaces. So, the interconnection of micro- and macroscale systems, into systems of systems, will accelerate.

It is not much of an exaggeration to say that in addition to our own formal disciplines we are all becoming systems scientists and engineers.^a

GEORGE HURLBURT

U.S. Federal Service (Retired), USA;
gfhurlburt@gmail.com
SOREL REISMAN California State University, USA;
sreisman@computer.org

EDITORS



The second marked trend, data-driven computing, is the defining aspect of this new era. Large and small ML compute architectures have provided entirely new and highly flexible ways to approximate and predict nonlinear functions and dynamic processes. Escaping linear and other explicit modeling techniques, the ML Era is especially delineated by the ability to represent, manipulate, and generate naturally occurring signals and information (text, speech, vision). However, this remarkable advance in learned computing results in systems for which there are no predictive theories, and progress is generally measured by empirical tests and comparisons.^b

TEAMS AND TESTBEDS

These two trends have opened doors to intelligent systems R&D. With data-driven elements, progress has a critical dependence on experimental cycles and spiral development. Virtual and physical testbeds provide the proving grounds, combining “real” data collection and simulations at various levels of abstraction.

Developing and evolving new ML system architectures requires flexible software modularity and careful repository management, with the ability to make surgical alterations and testing from the module to the larger system context, to alter and rearrange the system modules, and to explore various forms of feedback and intrasystem interactions.

Exhaustive tests do not scale, so ablation studies and other targeted statistical test measures are needed.^c

^bConsider a comparison with, say, designing a radar. The physics of electromagnetic propagation and object interaction, and radar signal design, provide an accurate predictive model that enables optimal design criteria and tradeoffs. However, using radar sensors in intelligent systems with data-driven learning (for example, driverless cars, cognitive radars) leans on empirical testing and evaluation.

^cWell-crafted ablation studies are the norm in ML research, as are a variety of test statistics and measures of goodness.

It is increasingly easy to pull together open source software to create a new system, and it is increasingly difficult to fully characterize system safety and reliability. The lack of predictive systems theory forces reliance on testing and red teaming and creation of system monitors. Even swapping out a single model-based module (for example, physics-based) for a data-driven one leads to test and verification challenges. Implicit or explicit system checking relies on

not easy. Although ML tools^e are now taught routinely within the various disciplines, there remains a “concept gap” between disciplines.^f Computer education can prepare students by offering cross-discipline coursework, collaborative opportunities, internship placement, double majoring, and university centers focused on major application areas. Of course, these are the bread and butter of flexible higher education.

Looking forward, it is interesting to think about deeper links with specific

Students need to be prepared for not just creating safe and reliable intelligent systems, but also how to monitor in situ and gauge their ongoing benefits and interactions.

carefully constructed critics, rules, boundaries, and constraints. Test, verification, and safety will be supported by a growing toolbox that is adapted to the case at hand, so education may focus on gaining familiarity with the tools and preparing students for their use in new contexts.^d

Intelligent systems development requires both compute and domain expertise. Multidisciplinary teams now readily combine two or more of computer science, any form of engineering, math/stat, linguistics, biology, neuroscience, psychology/sociology, agronomy, and more. This brings a cross-specialty burden that often lands on the computing specialist, requiring sufficient knowledge in a new domain.

Each area of science has evolved its own semantics and core concepts, and building a new mental model is

^dContinuing the radar example, the physics enable constraint and resource specifications that can be folded into learning with great benefit. However, this is specific to the application and doesn’t provide a predictive theory for, say, learning-based cognitive radar.

disciplines and how these may be incorporated into core computer education curricula, such as learning and cognitive science, neuroscience, and autonomy. These are natural combinations, and they naturally lead to systems.

INTELLIGENT FOUNDATION COMPUTING

With expansive ML Era compute centers come massive foundation models^g and new educational challenges. Training large scale models is a demanding task, with layers of pretraining, fine-tuning, and interacting with experts (for example, preference optimization). While remarkably general purpose, large models provide some

^eSoftware tools, libraries, and “platforms” have blossomed in the ML Era. The successful student will gain skills in the understanding and application of tools, up to modifying and creating new ones.

^fPerhaps we might appeal to AI for a useful cross-disciplinary translator just as we might hope, for example, that an AI could smooth patient–physician interactions.

^gIt seems remarkable that the term “foundation model,” attributed to a 2021 report from Stanford, is so new and yet already so fundamental.

TABLE 1. Intelligent systems combine key elements of classical system theory and ML-based intelligent computing.

State space representation	<ul style="list-style-type: none"> • Discrete and continuous state-space • State-action • Markov decision processes • Model-based and learning-based
Architecture and connectivity	<ul style="list-style-type: none"> • Centralized-decentralized • Synchronous-asynchronous • Hierarchical, graphs, trees • Distributed, worker-server • Feedforward and feedback • Memory and computing resources
Networking	<ul style="list-style-type: none"> • Bandwidth • Routing • Delay • Reliability
Learning	<ul style="list-style-type: none"> • Information representation and abstraction • Tokens and attention • Semantic and neuro-symbolic • Prediction and generation • Encoding-decoding • Explore-exploit • Games • Resource allocation • Optimization (multiobjective, distributed, dynamic, stochastic) • Gradient descent
Metrics	<ul style="list-style-type: none"> • Value, cost, reward, error, and utility functions • Measures of uncertainty • (Formal) model checking
Safety and Security	<ul style="list-style-type: none"> • Privacy and data protection • Constraints • In situ monitoring • Resilience to deception and attack

form of best estimate or statistical prediction.^h Adapting or improving foundation model response may rely on additional external processing, such as feedback-based query refinement, output bias detection and elimination, and use of external memory such as retrieval-augmented generation.

When broadly trained, a foundation model may respond in ways that are unacceptable (for example, for safety or ethical reasons), so input filters (for

example, query classification) can be used to avoid undesired prompts.

Foundation models may form a backbone that interacts with supporting models or headsⁱ to create an intelligent system, for example, a driverless car autonomy with heads that provide navigation plans, perception, object motion prediction, and so on.^j This enables separate creation, evolution, and improvement of the specific

purpose models within the system context.

System management and user interaction are now relying on foundation models, and these will blend with traditional operating systems. Foundation models are viewed as a potential new hybrid form of operating system, for example, using large language models, providing the user with semantic interaction and linking with tools, generators, and solvers. These will influence and coevolve with specialized integrated circuits designed for mass market computing (for example, smart phones, laptops).^k

The educational challenge is to understand various forms of foundation model training and refinement, querying, measures of risk and reward, human interaction, and how a foundation model can connect with and draw information from heads to serve a specific application. This, combined with a solid grasp of digital era operating systems, will prepare the student for next-gen intelligent foundation computing.

NETWORKED INTELLIGENT COMPUTING

At ultra-scale, the energy and communications resource consumption of foundation models has become very painfully evident, further motivating smaller model approximations that achieve a desired performance-complexity tradeoff. Connecting at a distance is good news (on-demand availability of a variety of models and applications) and bad news (cost, delay, bandwidth limitations).

We can expect a growing proliferation of models in all shapes and sizes, deployed at both micro- and macro-scale. Over time, what is commonly called a “big” model will evolve, and

^hIn some cases, such as large language models, generation of content not based on fact is undesirable and referred to as a “hallucination.” In other cases, a large model may be used for artistic generation (for example, music, imagery), and novelty is sought after.

ⁱNow used more generally, head originally refers to the top (that is, output or last) layer in a neural network, such as a classification layer in a convolutional neural network. ^jWe can think of this as a single-agent foundation (backbone) model that is capable of interaction and reasoning over various sources of information, evidence, planning, and prediction provided by additional modules.

^kState-of-the-art dense integrated circuit development and manufacturing is very expensive and so relatively few new devices are generated, and these are typically aimed at large scale applications. Analog computing alternatives have classically struggled with accuracy, programmability, and the need for (digital) calibration, yet may offer much higher speed and lower power. Forms of probabilistic and neuromorphic computing potentially handle these issues.

models with billions of parameters (or far more, depending on the future time scale we care to predict) will become even more commonplace and embedded into microscale devices.¹ With communications, these combine into networked computing and heterogeneous multiagent systems.²

From micro to macro, edge to cloud, computing networks rely on resource allocation that includes communications bandwidth and quality of service, power and energy, and available compute resources. This also includes data representation, processing, and storage, for example, to locally pre-store data in a consumer on-demand multimedia application, or compactly represent information optimized for learning an edge task.

When coupled with multiple concurrent tasks or users sharing a network, resource allocation becomes a challenging dynamic multiobjective optimization problem. Widely deployed compute networks, such as controlling the power grid or the Internet of Things, are complex systems of systems, for example, building on cellular systems and the Internet, each of which is highly complex and dynamic.

With stationary demand and dedicated centralized networking infrastructure, a specific multiobjective resource optimization along a Pareto front may be possible. This is more often an ideal and unachievable goal, with evolving real-time conditions requiring abstractions for robust dynamically updated allocations to steer tradeoffs and achieve overall performance metrics. Beyond classical optimization, introducing students to dynamic multiobjective methods is desirable.

¹The continued evolution of memory and computing circuits will accommodate larger models at micro-scale.

²We might think of a computer network as a (relatively static) form of distributed computing, whereas a multiagent system consists of independent and collaborating agents with mixed goals. However, there is a spectrum of possibilities, and the definitions are not rigid.

ML SYSTEMS PRINCIPLES

ML Era intelligent systems are a blend of classic systems and intelligent computing; see Table 1. Learning-based representation and abstraction, coupled with architecture and connectivity, provide a rich systems design space.

In particular, graph-based architectures are remarkably versatile and can be understood intuitively. They fit communications networks and underlie multiagent interactions. They model the physical world through geometric-semantic maps, enable planning and navigation, and generalize beyond two dimensions. They naturally express dependence and directed graphs model asymmetry (for example, causality³). Graphs also provide a powerful inference architecture, for example, graph neural networks. An educational challenge is not whether graphs are fundamentally important, but rather how to avoid confusion among all the many ways they may be employed.

MONITORING INTELLIGENCE

It is apparent that intelligent systems offer much and threaten many. As the ML Era progresses, in various ways, computing will become synonymous with intelligence.

Open source, powerful edge devices, and cloud computing availability all add up to consumer opportunity. Digital playgrounds allow any of us to teach robots to walk and create new interactive artificial intelligence (AI) agents.

We've argued that education should include creating and applying tools for monitoring, safety, test and evaluation. However, a larger challenge is to educate students on the issues relating to use and application, risks and rewards, and societal gains and losses. Students need to be prepared for not

³Causality is easy to intuitively understand, but difficult to formally characterize. A graph provides a structure over which causality and other forms of dependence can be learned.

just creating safe and reliable intelligent systems, but also how to monitor in situ and gauge their ongoing benefits and interactions.

General computer education will include secure computing, privacy, and data protection. These are greatly magnified with intelligent computing, and the importance cannot be overstated. These should not be viewed as add-ons, to be taught separately. Rather, they need to be an integral part of any intelligent system, including resilience to adversarial attack. It is our responsibility to embrace this into all aspects of computer education.⁴

As the ML Era progresses, the expected baseline knowledge and core curricula will distill and mature. Many of these concepts become apparent through in-depth study of systems and examples. It is also important to instill the underlying ideas and build understanding. With a foundation in rudiments, the student can mature into an intelligent computing maestro.⁵

⁴Just as in communications and network engineering education, where security is inherent and inseparable, similar ideas should be part of the culture of computer education.

⁵Beginning percussion students (such as the author, many years ago) often learn drum rudiments, including various drum rolls, diddles, drags, ruffs, and flams. For example, there are 26 U.S. Standard Rudiments, defined in 1933 and commonly taught today. These are short sticking patterns that can be combined to build formal drum parts (for example, orchestral, marching). The rudiments span a rich set of subtle coordination and mastering them provides a solid technical baseline.

BRIAN M. SADLER is a senior research fellow at the University of Texas at Austin in the Oden Institute for Computational Engineering and Sciences, Austin, TX 78712 USA. Contact him at brian.sadler@ieee.org.



Blockchain Standardization in Practice: Contrasting European Union and U.S. Approaches

Nir Kshetri¹, The University of North Carolina at Greensboro

This article categorizes blockchain standards by their functional focus and by how they are established. It also contrasts the European Union's and the United States' regulatory approaches in blockchain standardization in key areas.

The 2024 valuation of the global blockchain market at US\$31 billion¹ underscores its potential, yet the lack of standardization continues to hinder its broader adoption. For instance, by 2025, the sectoral visibility of blockchain initiatives in Canada, particularly those extending beyond cryptocurrency

applications, has experienced a measurable decline in prominence relative to prepandemic levels. Efforts by firms like Walmart Canada and major banks to implement blockchain in payments and supply chains have faced delays, primarily due to challenges in achieving the necessary standardization.²

Standards are essential for global interoperability and market flexibility, facilitating seamless cross-blockchain data exchange. However, persistent fragmentation in standardization efforts remains a barrier.³ The inherent complexity of blockchain

and related technologies, spanning diverse technical, regulatory, and operational domains, requires collaboration among diverse stakeholders, complicating both the development of unified standards and the evolution of current standardization ecosystems.⁴

The role of standards in distributed ledger technology (DLT) and blockchain is thus widely acknowledged. However, views vary on specific areas for standardization and implementation timelines.⁵

Digital Object Identifier 10.1109/MC.2025.3563736
Date of current version: 27 June 2025



Blockchain requires standards for various key areas, including interoperability for seamless communication among platforms, governance to manage decentralized projects, identity frameworks for consistent identity protocols, security to protect networks and nodes, and best practices for ensuring the safety of smart contracts. These standards are crucial for creating a robust, secure, and cohesive blockchain ecosystem.⁶

Interoperability is challenging due to variations in technology, standards, and legislation. The absence of a global standard for blockchain-based digital identification leads to interoperability issues, hindering system integration and slowing adoption.⁷

The development of technological standards is shaped by a confluence of technical, commercial, political, and moral imperatives.⁸ While market dynamics and regulatory interventions influence their adoption, scholarly analysis underscores the pivotal role of state actors: standards backed by governmental mandates exhibit a higher likelihood of market dominance.⁹ Governments, leveraging regulatory

authority, strategically steer standardization processes to align with national economic and technological objectives.¹⁰

Government regulation can play a pivotal role in unifying fragmented blockchain standards. By leveraging their authority, policy makers can drive the development of consistent frameworks that enhance interoperability, close regulatory and technical gaps, and improve data security across blockchain networks. In the absence of such coordination, developers may continue to adopt divergent or incomplete standards, exacerbating fragmentation. A unified regulatory approach, requiring intergovernmental collaboration, clear leadership, and dedicated resources, can promote more secure and interoperable blockchain ecosystems across sectors.¹¹

This article examines blockchain standards by their functional focus and by the processes through which they are established. It also compares the regulatory roles in key standardization areas between the European Union (EU) and the United States.

TAXONOMY OF BLOCKCHAIN STANDARDS: TYPES AND CATEGORIES SHAPING INDUSTRY PRACTICES

Standards exist in various types and categories, each serving distinct purposes in blockchain systems. They ensure consistency, reliability, and interoperability, guiding the design, implementation, and evaluation of blockchain applications across industries. This section categorizes standards based on their functional focus and how they are developed and adopted.

Types of standards based on their functional focus

Standards can be classified based on their functional focus, what they aim to define, measure, or enable. In the blockchain ecosystem, these include measure or metric standards, process-oriented or prescriptive standards, performance-based standards, and interoperability standards (Table 1). Each type plays a distinct role in promoting consistency, reliability, and compatibility across blockchain systems and applications, supporting the technology's

TABLE 1. Types of standards and their applications in blockchain systems.

Type	Definition and characteristics	Examples/applications in blockchain
Measure or metric	Reference points used to quantify and compare attributes. These standards enable consistency in measurement and facilitate informed decision making.	Ethereum's "gas" for computational effort in smart contracts, priced in gwei and fluctuating based on network demand.
Process oriented	Provide structured procedures and best practices to ensure repeatability and consistency. These are often regulatory or compliance oriented.	Financial Action Task Force standards for virtual assets and virtual asset service providers (for example, anti-money-laundering/countering the financing of terrorism compliance, licensing requirements).
Performance based	Focus on the outcomes rather than specific methods, offering flexibility in how results are achieved. These standards prioritize final objectives like security, control, and privacy.	EU Digital Identity Framework (eIDAS 2) supports blockchain-based IDs for security and cross-border recognition.
Interoperability	Ensure systems can communicate and operate together by using common formats, without dictating internal processes or performance levels. These standards promote compatibility across platforms and services.	Chainlink's corporate actions data standardization project in Europe; the use of artificial intelligence and oracles (ChatGPT, Gemini, and Claude) to create structured Golden Records compliant with ISO.

scalability and integration across industries.

A measure or metric standard is a reference against which comparable quantities are measured. Examples include the kilogram for mass, the meter for length, and the liter for volume. These standards are particularly beneficial for consumers as they facilitate comparison shopping for price, function, or features.¹² Ethereum uses gas to measure the computational effort for executing smart contract operations. Gas costs depend on the complexity and resource usage of the operation. Prices are denominated in gwei (a fraction of Ether) and fluctuate based on network demand.¹³

Process-oriented standards provide structured guidelines for executing tasks in a consistent and reproducible manner.¹² The Financial Action Task Force (FATF) is an intergovernmental body established in 1989 by the G7 to set global standards for combating money laundering. Since 2001, its mandate has expanded to include countering terrorist financing. In 2019, FATF updated its standards on virtual assets (VAs) and virtual asset service providers (VASPs), followed by a comprehensive review in 2020. The FATF standards involve a number of recommendations that provide a comprehensive framework for combating money laundering and terrorist financing in the cryptocurrency sector. For instance, under the amended FATF Recommendation 15, VASPs must be regulated, licensed, or registered and implement anti-money-laundering (AML)/countering the financing of terrorism (CFT) measures similar to traditional financial institutions. VASPs are required to gather and transmit sender and recipient details for transactions exceeding a specified threshold to maintain traceability and regulatory compliance. Countries must regulate and monitor VASPs to enforce AML/CFT measures and ensure compliance with FATF recommendations, mitigating money laundering and terrorism financing risks.¹⁴ This type

of standard is process oriented or prescriptive, standardizing activities and methodologies to ensure consistency and repeatability in testing and operations.

Performance-based standards focus on the final outcome rather than the processes involved. They specify the desired end result but leave flexibility in how to achieve it.¹² For instance, blockchain-based digital identities use a variety of performance measures related to security, privacy, and control. The EU Digital Identity Framework is built on three key pillars designed to enhance security, accessibility, and user control. The first pillar strengthens national electronic identification systems under electronic identification, authentication, and trust services (eIDAS), ensuring cross-border recognition across EU member states for smoother identity verification. The second pillar involves the private sector, enabling companies to provide identity-linked services while adhering to eID regulations. The third pillar introduces the EU Digital Identity Wallet, a secure app that allows users to manage and control their identity data, ensuring privacy and portability. Blockchain plays a crucial role in supporting the framework, linking credentials to decentralized identifiers on the blockchain to ensure security and authenticity. The wallet employs biometric authentication for access, securely stores data, and provides users with full control over their information, enabling them to share only necessary details. Additionally, the wallet is designed for interoperability, ensuring seamless use across different services and EU member states.¹⁵

The final type of standard focuses on interoperability, where systems are required to work together seamlessly. These standards do not explicitly define processes or performance metrics but specify a fixed format to ensure smooth operation among systems using the same physical entity or data. The goal is to enable compatibility and coordination across different systems

without dictating how each should perform or function.¹² In the financial sector, Chainlink has launched an initiative to standardize and improve access to corporate actions data through artificial intelligence (AI) and blockchain, addressing the issue of fragmented information, especially in Europe. Corporate action data, such as dividends, mergers, and stock splits, often come in inconsistent formats, leading to errors and financial losses. Despite efforts by organizations like the Depository Trust & Clearing Corporation, standardizing these data has been an ongoing challenge. The initial phase of Chainlink's project focuses on equity and fixed-income securities in six European countries. It will use decentralized oracles and advanced AI models like OpenAI's ChatGPT, Google's Gemini, and Anthropic's Claude to extract and structure corporate actions data into standardized "Golden Records" that comply with international standards, such as ISO 20022. These structured data will be shared across multiple blockchains using Chainlink's Cross-Chain Interoperability Protocol (CCIP). This initiative is expected to reduce manual processes, improve operational efficiency, and cut costs.

Categories of standards based on establishment processes

Standards can also be categorized based on how they are developed and adopted, whether through market dynamics, regulatory mandates, or formalized collaboration. In the blockchain domain, all three categories, de facto, regulatory, and consensus standards, play critical roles in shaping the technology's evolution (Table 2). These standards not only guide blockchain development and deployment but also influence how ecosystems interoperate, gain legitimacy, and achieve mass adoption.

A de facto standard is widely accepted and used without formal approval, emerging through market consensus. Examples include the

QWERTY keyboard, PC architecture, and the UNIX operating system.¹² Ethereum can be viewed as a de facto standard in the blockchain industry, especially for smart contracts and decentralized applications (dApps). Flipside’s “EVM Smart Contract Deployment Snapshot” report indicates that 637.9 million Ethereum Virtual Machine smart contracts have been deployed from January 2022, within a little over two years.¹⁶ Likewise, as the largest blockchain oracle platform, Chainlink is focused on creating standards for blockchain oracles.¹⁷

Regulatory standards are established by agencies to ensure uniformity in processes independent of market forces.¹² As blockchain gains recognition, regulatory standards are evolving, with governments, international organizations, and regulators addressing its growing significance. The EU’s Markets in Crypto Assets (MiCA) Regulation is focused on creating clear rules for crypto assets, protecting investors, and ensuring that crypto service providers comply with consumer protection requirements. The European Blockchain Services Infrastructure (EBSI) initiative seeks to create technical standards that facilitate cross-border interoperability for blockchain applications in public services across the European Union.

Consensus standards are voluntary standards developed by domestic or international bodies using agreed-upon procedures. These standards are created by organizations that plan, develop, and coordinate voluntary standards.¹² As of 2023, at least

30 organizations, including IEEE and GS1, were developing separate or overlapping standards.³

For instance, IEEE P3222.01, *Standard for Blockchain-Based Digital Identity Systems*, defines requirements for blockchain-based digital identity systems, covering identity creation, authentication, credentials (for example, ID cards or work cards), and data circulation protocols. It has been active since May 2020.¹⁸

STANDARDIZING THE FUTURE: A TRANSATLANTIC PERSPECTIVE ON BLOCKCHAIN REGULATION

Europe and North America are key regions where standards-setting activities are predominantly concentrated,¹⁹ reflecting their strategic roles in shaping global blockchain interoperability, governance, and regulatory frameworks. In this regard, **Table 3** outlines the key areas where the European Commission (EC) considers blockchain standardization essential⁶ and compares the regulatory roles in these areas between the EU and the United States.

Interoperability

Blockchain interoperability is referred to as “the ability of blockchain networks to communicate with each other, sending and receiving messages, data, and tokens.”²⁰ Key challenges in blockchain include the systematic benchmarking of interoperability solutions. This involves a structured evaluation of various blockchain solutions to measure their performance, efficiency,

and compatibility across different networks, helping to identify the most effective solutions and areas for improvement. Additionally, there is a lack of standardized terminology as academia and industry often use different language, especially in rollups research.²¹

The EU drives blockchain interoperability through government-led infrastructure and regulatory alignment, while the United States relies on industry-driven pilots and sector-specific standards. The commission collaborates with the private sector, academia, and the blockchain community through the International Association of Trusted Blockchain Applications, a public/private partnership that promotes blockchain interoperability and governance and serves as a liaison with governments and international bodies.²² The European public sector is creating its own blockchain infrastructure, which will soon be interoperable with private sector platforms. The EBSI is a peer-to-peer network of nodes run by the 27 EU countries, Norway, Liechtenstein, and the EC. It includes a base layer for infrastructure and storage, a core services layer for EBSI applications, and additional layers for specific use cases. The infrastructure will enable public organizations to develop applications, with plans to extend it to private organizations.²³ EBSI aims to provide a shared, secure, and interoperable infrastructure for EU-wide cross-border public sector digital services, reflecting European values like data sovereignty and sustainability. It will address global issues such as climate change and supply

TABLE 2. Categories of standards and their applications in the blockchain ecosystem.

Category	Definition	Blockchain example
De facto standard	Widely adopted through market consensus without formal approval.	Ethereum for smart contracts and dApps; Chainlink as the leading oracle network.
Regulatory standard	Set by governmental or intergovernmental agencies to ensure legal compliance.	EU’s MiCA Regulation for crypto asset oversight; EBSI for public service blockchain interoperability.
Consensus standard	Voluntary standards developed through collaborative, agreed-upon processes.	IEEE P3222.01 for blockchain-based digital identity systems.

chain corruption, while ensuring high standards of scalability, security, and privacy. The infrastructure should be deployed within three years. Built as a “public permissioned” blockchain, EBSI’s interoperable peer-to-peer network consists of 36 live nodes, with 11 more in setup, managed by the EC and EU member states.²⁴

U.S. federal government initiatives, led by agencies such as the Department of Homeland Security (DHS), the U.S. Customs and Border Protection (CBP), the Department of the Treasury, and the Government Accountability Office, have aimed to advance blockchain interoperability, emphasizing operational applications, interagency collaboration, and the development of common standards. Pilot projects have tested blockchain’s ability to streamline data sharing and

verification processes across agencies. For example, the CBP within the DHS explored blockchain’s potential to improve trade documentation and verify import legality, highlighting benefits such as enhanced interoperability and data integrity.²⁵ The Treasury and GAO expanded a blockchain prototype to a two-agency network under the JFMIP, emphasizing the importance of shared services and interoperability testing.³ DHS’s S&T Directorate, through its Silicon Valley Innovation Program, has worked with startups to develop interoperable standards for supply chain security and digital credentialing.²⁶

U.S. federal agencies are collaborating with the private sector to improve blockchain interoperability, particularly in complex pharmaceutical supply chains. Current blockchain

solutions, while industry specific, lack interoperability, creating challenges for firms adopting different systems to conduct business. As part of the FDA’s program to evaluate the use of blockchain to protect pharmaceutical product integrity, Merck and Walmart partnered with IBM and KPMG in the DSCSA Pilot Project Program under section 582(j) of the FD&C Act in March 2019. The initiative aimed to assess blockchain’s potential in ensuring interoperability among trading partners and meeting DSCSA 2023 compliance requirements. The project also explored blockchain’s value beyond compliance, particularly in improving the medication recall process.²⁷

Governance

Countries are revising regulatory frameworks to attract crypto businesses,

TABLE 3. Key areas in blockchain standardization.

Area	Explanation	EU	United States
Interoperability	Enabling seamless data exchange and communication among different blockchain and DLT platforms.	The public sector is creating its blockchain infrastructure, which will be interoperable with private sector platforms	Federal agencies’ initiatives to advance interoperability, emphasizing operational applications, interagency collaboration, and the development of common standards. Federal agencies collaborating with the private sector to improve blockchain interoperability.
Governance	Setting rules, processes, and guidelines for managing blockchain projects and consortia on decentralized platforms.	MiCA aims to provide regulatory clarity for crypto assets and consumer/investor protection. No specific regulations for decentralized autonomous organizations (DAOs).	Federal regulation of cryptocurrencies and DAOs pending. States like Wyoming have recognized DAOs.
Identity	Establishing a unified or compatible identity system across various blockchain protocols and platforms.	eIDAS 2 explicitly recognizes DLT-based electronic trust services, granting them the same legal status as traditional services.	No national standard for digital identity: Utah was the first state to integrate blockchain into digital identity management.
Security	Maintaining the safety and reliability of nodes, networks, and services.	MiCA sets out requirements for blockchain nodes to reduce transaction risks and protect network participants.	The CETU’s focus on dark web investigations, cryptocurrency fraud, and blockchain-related crimes.
Smart contract	Establishing guidelines and standards to enhance the security and reliability of smart contract technology.	MiCA regulation lacks full smart contract provisions. 2024: France’s Autorité de contrôle prudentiel et de résolution collaborating with industry to mandate the certification of smart contracts before use.	The United States relies on varying state laws without a unified federal approach.

with a focus on governance standards and investor protection.²⁸ The EU's MiCA Regulation establishes uniform rules for unregulated crypto assets, emphasizing governance through transparency, disclosure, authorization, and oversight to enhance market integrity, financial stability, and consumer protection.²⁹

In the United States, regulatory uncertainty persists as lawmakers and industry stakeholders debate whether the Securities and Exchange Commission (SEC) or the Commodity Futures Trading Commission (CFTC) should oversee the crypto market. This ongoing disagreement, rooted in whether crypto assets are classified as securities or commodities, reflects broader governance challenges in establishing a clear and consistent regulatory framework.³⁰ The SEC is taking a stricter stance on proof-of-stake (PoS) tokens than proof-of-work assets. Chair Gary Gensler has suggested PoS tokens may be securities under the Howey test as they involve profit expectations based on others' efforts.³¹

One key area of blockchain governance is decentralized autonomous organizations (DAOs), which rely on smart contracts and token-based participation to enable decentralized decision making. In the EU, while MiCA provides regulatory clarity for crypto assets and enhances consumer and investor protection, it does not specifically address the distinctive governance structures and legal status of DAOs.³² In the United States, while federal regulation of cryptocurrencies and DAOs remains pending, states like Wyoming have proactively recognized DAOs as a form of limited liability company, providing a legal framework for their operation.³³

Identity frameworks

Blockchain identity systems rely on advanced technologies and standards to ensure security, privacy, and user control. They incorporate key components that form a robust and reliable

framework for managing digital identities.³⁴

The EU's eIDAS 2 regulation establishes legal standards for DLT-based electronic trust services, enabling interoperability and removing key barriers to blockchain adoption.³⁵ By granting blockchain-based services the same legal status as traditional ones, it promotes integration into regulated sectors; supports smart contract enforceability; and encourages innovation across industries such as finance, real estate, and energy.³⁶

In the United States, the absence of a national digital identity standard has led the federal government to delegate much of the responsibility to individual states.³⁷ Some states are developing regulatory frameworks for digital identity based on blockchain. These frameworks aim to enhance security, privacy, and control over personal data, offering a more transparent and decentralized approach to managing digital identities. Utah Governor Spencer Cox recently signed HB 470, mandating the state's Division of Technology Services (DTS) to launch a pilot program for digital verifiable credentials using blockchain. The bill requires DTS to provide recommendations on issuing digital IDs or records through DLT, as well as policies to protect personal privacy.³⁸ Utah was the first state to integrate blockchain technology into digital identity management.³⁹ Utah demonstrated its crypto ambitions by becoming one of the first states to accept digital assets for certain payments, including local and state taxes, placing it among a select few states taking this step.³⁹

Cybersecurity

Regulators in the EU and the United States are taking measures to enhance blockchain security by focusing on maintaining the safety and reliability of nodes, networks, and services. This includes implementing stricter compliance requirements, oversight, and security standards to ensure that blockchain systems operate securely

and are resilient against potential threats.

The European Securities and Markets Authority (ESMA) has recommended that MiCA include mandatory third-party cybersecurity assessments for crypto firms and establish consistent security protocols across the EU. ESMA contends that FTX's collapse underscores the need for rigorous cybersecurity audits to strengthen crypto company resilience, although the EC cautions that such measures might exceed MiCA's intended scope.⁴⁰ MiCA sets out requirements for blockchain nodes to reduce transaction risks and protect network participants. Node operators offering commercial services must register with EU regulators and disclose details about their operations, infrastructure, and risks to enhance transparency. Nodes must implement strong data security measures, including encryption and backup, especially when handling sensitive data or high transaction volumes. Additionally, operators must follow AML and know-your-customer procedures to prevent illegal activities. Node operators are legally responsible for complying with these regulations and may face sanctions or fines for noncompliance.⁴¹

In the United States, the SEC is focusing on cybersecurity involving crypto assets to address risks posed to investors. Focused on cybersecurity and innovation oversight, the SEC's Cyber and Emerging Technologies Unit (CETU) investigates bad actors exploiting emerging technologies to deceive retail investors.⁴² The SEC has also replaced its Crypto Assets and Cyber Unit with the new CETU. The CETU will focus on dark web investigations, cryptocurrency fraud, and blockchain-related crimes, reflecting the growing government and public attention on cryptocurrency.⁴³

Smart contracts

Smart contracts are computer programs that produce sequences of bits but do not define their meaning or

correct interpretation. For instance, a sequence like “e, s, t, a, t, e” could represent “estate,” but it might also be random data, and the term “estate” has different meanings in different languages. Thus, smart contracts require external standards to properly encode/decode data and guide interpretation. These rules cannot be stored on the blockchain itself, as that would create a circular problem.⁴⁴ For smart contracts to function effectively, standards are thus key, particularly given the presence of users in multiple jurisdictions with different languages.

Smart contracts run exactly as coded, leaving no room for error; once deployed, they cannot be fixed, only replaced with a new version, which is costly and time consuming. Smart contract auditors are essential for ensuring the code’s safety and security.⁴⁵ Therefore, establishing robust standards for smart contract development and auditing is crucial to ensure their reliability, security, and efficiency throughout their lifecycle.

Both government agencies and the private sector play vital roles in the implementation of auditing standards, ensuring compliance, promoting best practices, and enhancing the overall security of smart contracts. The Cardano Smart Contract Certification program sets standards for auditing and certifying smart contracts on Cardano, enhancing security and reliability through formal verification and building confidence among users and developers.⁴⁶

The EU's MiCA regulation provides a broad blockchain framework but lacks full smart contract provisions.⁴⁷ Individual EU countries are also advancing smart contract regulation. In 2024, France's Autorité de contrôle prudentiel et de résolution (Prudential Supervision and Resolution Authority), with support from the Banque de France, proposed certifying smart contracts before deployment to ensure security and consumer protection. The initiative, which includes regulating decentralized finance platforms and

blockchain infrastructure, reflects France's broader influence on EU-level crypto policy and aims to balance innovation with risk mitigation.⁴⁸

The United States relies on varying state laws without a unified federal approach.⁴⁷ In 2017, Arizona became the first U.S. state to recognize smart contracts by passing legislation that included blockchain-based signatures and records. Tennessee followed in 2018, amending its statutes on electronic forms and signatures to incorporate blockchain.⁴⁹ In 2020, Illinois enacted the Blockchain Technology Act, which defines and enforces smart contracts under specified conditions. New York also introduced a bill that recognizes the use of smart contracts, although it is limited to commercial transactions.⁵⁰

The contrasting approaches to blockchain standardization between the EU and the United States underscore the critical role of regulatory frameworks and collaborative governance in overcoming fragmentation and fostering global adoption. While the EU has leveraged centralized, principle-based initiatives like MiCA and EBSI to drive interoperability, security, and legal clarity across member states, the United States has adopted a decentralized model reliant on industry innovation and state-level experimentation. Both regions face persistent challenges, such as reconciling blockchain's decentralized nature with compliance requirements, addressing interoperability gaps, and harmonizing technical standards, that demand robust frameworks spanning metric- and performance-based paradigms. Moving forward, bridging transatlantic disparities through international collaboration on consensus standards, shared security protocols, and interoperable identity systems will be essential to unlocking blockchain's full potential as a scalable, secure, and globally integrated technology. □

REFERENCES

- “Blockchain technology market size, share & trends analysis report,” Grand View Res., San Francisco, CA, USA, Rep. ID: GVR-1-68038-329-4, Jan. 2025. Accessed: Apr. 19, 2025. [Online]. Available: <https://www.grandviewresearch.com/industry-analysis/blockchain-technology-market>
- K. Stiff, “From real estate to identity standards, momentum rebuilds in Canada's blockchain industry,” *The Globe and Mail*, Feb. 15, 2025. Accessed: Apr. 19, 2025. [Online]. Available: <https://www.theglobeandmail.com/business/article-from-real-estate-to-identity-standards-momentum-rebuilds-in-canadas/>
- “Harnessing blockchain in the federal government: Key considerations for financial management and information systems,” Joint Financial Management Improvement Program (JFIMP), Dec. 2023. Accessed: Apr. 19, 2025. [Online]. Available: <https://www.cfo.gov/assets/files/JFIMP-24-01.pdf>
- C. Koch and K. Blind, “Towards agile standardization: Testbeds in support of standardization for the IIoT,” *IEEE Trans. Eng. Manag.*, vol. 68, no. 1, pp. 59–74, Feb. 2021, doi: [10.1109/TEM.2020.2979697](https://doi.org/10.1109/TEM.2020.2979697).
- “The potential role of standards in supporting the growth of distributed ledger technologies/blockchain,” RAND Europe, Cambridge, U.K. Accessed: Apr. 19, 2025. [Online]. Available: <https://www.rand.org/randeurope/research/projects/2017/blockchain-standards.html>
- European Commission. “Blockchain standards.” Digital Strategy. Accessed: Feb. 17, 2025. [Online]. Available: <https://digital-strategy.ec.europa.eu/en/policies/blockchain-standards>
- “Blockchain-based digital identity: Benefits, risks, and implementation challenges,” Finance Magnates, Limassol, Cyprus, 2023. Accessed: Apr. 19, 2025. [Online]. Available: <https://www.financemagnates.com/cryptocurrency/education-centre/>

- blockchain-based-digital-identity
-benefits-risks-and-implementation
-challenges/
8. R. Werle and E. Iversen, "Promoting legitimacy in technical standardization," *Sci. Technol. Innov. Stud.*, vol. 2, no. 1, pp. 19–39, 2006.
 9. D. J. Teece, "Capturing value from knowledge assets: The new economy, markets, know-how, and intangible assets," *California Manage. Rev.*, vol. 40, no. 3, pp. 55–79, 1998, doi: [10.2307/41165943](https://doi.org/10.2307/41165943).
 10. I. MacInnes, "A model for standard setting: High definition television," *Contemporary Econ. Policy*, vol. 12, no. 4, pp. 67–78, 1994, doi: [10.1111/j.1465-7287.1994.tb00446.x](https://doi.org/10.1111/j.1465-7287.1994.tb00446.x).
 11. "Blockchain: Emerging technology offers benefits for some applications but faces challenges," U.S. Government Accountability Office (GAO), Washington, DC, USA, GAO-22-104625, Mar. 2022. Accessed: Apr. 19, 2025. [Online]. Available: <https://www.gao.gov/assets/gao-22-104625.pdf>
 12. R. H. Allen and R. D. Sriram, "The role of standards in innovation," *Technol. Forecasting Social Change*, vol. 64, nos. 2–3, pp. 171–181, Jun. 2000.
 13. Buidly, "Gas costs: EVM & Non-EVM blockchains," *Medium*, Oct. 14, 2024. Accessed: Mar. 18, 2025. [Online]. Available: <https://medium.com/@buidly.tech/gas-costs-evm-non-evm-blockchains-f44daf3488d0>
 14. "12-month review virtual assets and VASPs," FATF, Paris, France, 2020. Accessed: Apr. 19, 2025. [Online]. Available: <https://www.fatf-gafi.org/publications/fatfrecommendations/documents/12-month-review-virtual-assets-vasps.html>
 15. Astrakode, "Digital identity and blockchain: Exploring the EU digital identity framework," *Medium*, Accessed: Apr. 19, 2025. [Online]. Available: <https://astrakode.medium.com/digital-identity-and-blockchain-exploring-the-eu-digital-identity-framework-cdec759e03ba>
 16. G. Matos, "EVM chains see over 637 million smart contracts deployed since Jan 2022: Flipside," *Cryptobriefing*, Mar. 8, 2024. Accessed: Mar. 18, 2025. [Online]. Available: <https://cryptobriefing.com/evm-smart-contract-deployment-surge/>
 17. G. Weston, "Top 10 blockchain oracles," Blockchain Georgia, Feb. 22, 2023. Accessed: Mar. 18, 2025. [Online]. Available: <https://101blockchains.com/top-blockchain-oracles/>
 18. IEEE Standard for Blockchain-Based Digital Identity Management, IEEE Standard 3222.01, 2024. Accessed: Apr. 19, 2025. [Online]. Available: <https://standards.ieee.org/ieee/3222.01/10242/>
 19. X. Jia, J. Xu, M. Han, Q. Zhang, L. Zhang, and X. Chen, "International standardization of blockchain and distributed ledger technology: Overlaps, gaps and challenges," *Comput. Model. Eng. Sci.*, vol. 137, no. 2, pp. 1491–1523, Jun. 2023, doi: [10.32604/cmes.2023.026357](https://doi.org/10.32604/cmes.2023.026357).
 20. "Interoperability report," Eur. Blockchain Observatory and Forum, Nov. 30, 2023. Accessed: Apr. 19, 2025. [Online]. Available: https://blockchain-observatory.ec.europa.eu/document/download/c289f656-052a-4a72-b213-26b307691844_en?filename=EUBOF_Interoperability%20Report-30112023.pdf
 21. R. Belchior, J. Süßenguth, Q. Feng, T. Hardjono, A. Vasconcelos, and M. Correia, "A brief history of blockchain interoperability," *Commun. ACM*, Sep. 24, 2024. Accessed: Apr. 19, 2025. [Online]. Available: <https://cacm.acm.org/research/a-brief-history-of-blockchain-interoperability/>
 22. European Commission, "Blockchain strategy: Shaping Europe's digital future," Digital Strategy. Accessed: Apr. 19, 2025. [Online]. Available: <https://digital-strategy.ec.europa.eu/en/policies/blockchain-strategy>
 23. European Commission, "European blockchain services infrastructure," Digital Strategy. Accessed: Apr. 19, 2025. [Online]. Available: <https://digital-strategy.ec.europa.eu/en/policies/european-blockchain-services-infrastructure>
 24. C. R. W. De Meijer, "European blockchain services infrastructure (EBSI): The European way to get the most out of blockchain," *Finextra*, Mar. 5, 2024. Accessed: Apr. 19, 2025. [Online]. Available: <https://www.finextra.com/blogposting/20963/european-blockchain-services-infrastructure-ebsi-the-european-way-to-get-most-out-of-blockchain>
 25. "CBP leverages blockchain innovation to protect American business." U.S. Customs and Border Protection (.gov). Accessed: Feb. 28, 2025. [Online]. Available: <https://www.cbp.gov/newsroom/national-media-release/cbp-leverages-blockchain-innovation-protect-american-business#:~:text=The%20technology%20is%20critical%20to,rapid%20adoption%20and%20cost%20reduction>
 26. "Feature article: S&T's Silicon Valley innovation program leverages blockchain interoperability to support DHS," Dept. of Homeland Secur. (DHS), Washington, DC, USA, Oct. 8, 2020. Accessed: Apr. 19, 2025. [Online]. Available: <https://www.dhs.gov/science-and-technology/news/2020/10/08/feature-article-st-svip-leverages-blockchain-interoperability-support-dhs>
 27. "FDA DSCSA: Blockchain interoperability pilot project report," Feb. 2020. Accessed: Oct. 10, 2023. [Online]. Available: <https://www.fda.gov/media/169883/download>
 28. "How regulation will shape key crypto narratives in 2025." Mercuryo. Accessed: Apr. 19, 2025. [Online]. Available: <https://mercuryo.io/explore/article/how-regulation-will-shape-key-crypto-narratives-in-2025>
 29. "Markets in crypto-assets regulation (MiCA)." European Securities and Markets Authority (ESMA). [Online]. Accessed: Oct. 10, 2023. Available: <https://www.esma.europa.eu/esmas-activities/digital-finance-and-innovation/markets-crypto-assets-regulation-mica>
 30. A. Kumar. "What is next for crypto regulation in the US?"

- CryptoPolicyReview.com. Accessed: Oct. 10, 2023. [Online]. Available: <https://www.atlanticcouncil.org/blogs/econographics/what-is-next-for-crypto-regulation-in-the-us/>
31. J. Valente, "Clash of consensus: How the SEC's stance on proof of stake tokens challenges Crypto's Green Future," *Georgetown Environmental Law Rev.*, Apr. 17, 2024. Accessed: Apr. 19, 2025. [Online]. Available: <https://www.law.georgetown.edu/environmental-law-review/blog/clash-of-consensus-how-the-secs-stance-on-proof-of-stake-tokens-challenges-cryptos-green-future/>
32. J. M. de Corral, "DAO legal landscape: An overview of challenges & approaches." *Rif.technology*. Accessed: Apr. 19, 2025. [Online]. Available: <https://rif.technology/content-hub/dao-regulations/>
33. P. Horbonos, "Introduction to digital identity blockchain," *Blaize Tech Blog*, Aug. 12, 2024. Accessed: Apr. 19, 2025. [Online]. Available: <https://blaize.tech/blog/blockchain-digital-identity/>
34. Jurisconsul. "EU's digital revolution: How eIDAS 2 could spark a blockchain boom in government and industry." *Lexology*. Accessed: Apr. 19, 2025. [Online]. Available: <https://www.lexology.com/library/detail.aspx?g=af833dbd-f83f-4358-8594-ecaa0e8306a8>
35. E. Sotiri, "EU's digital revolution: How EIDAS 2 could spark a blockchain boom in government and industry." *Jurisconsul*. Accessed: Apr. 19, 2025. Accessed: Apr. 20, 2025. [Online]. Available: <https://www.jurisconsul.com/post/eu-s-digital-revolution-how-eidas-2-could-spark-a-blockchain-boom-in-government-and-industry>
36. A. Johnson, "The path to digital identity in the United States," Inf. Technol. and Innov. Found., Washington, DC, USA, Sep. 23, 2024, Accessed: Apr. 19, 2025. [Online]. Available: <https://itif.org/publications/2024/09/23/path-to-digital-identity-in-the-united-states/>
37. K. Quinlan, "Utah governor signs blockchain digital ID pilot into law," *StateScoop*, Apr. 12, 2023. Accessed: Apr. 19, 2025. [Online]. Available: <https://statescoop.com/utah-governor-signs-blockchain-digital-id-pilot-into-law/>
38. B. Norton, "Navigating the legal framework: Implementing a government-backed digital identity in the United States," *Jurimetrics*, vol. 64, pp. 169–199, Winter 2024. Accessed: Apr. 19, 2025. [Online]. Available: <https://www.americanbar.org/content/dam/aba/publications/Jurimetrics/winter-2024/navigating-the-legal-framework-implementing-a-government-backed-.pdf>
39. D. Hamilton, "Top 10 pro-crypto states you need to know," *Securities.io*, Sep. 4, 2023. Accessed: Apr. 19, 2025. [Online]. Available: <https://www.securities.io/top-10-pro-crypto-states-you-need-to-know/>
40. "EU crypto firms face cyber audits and ESMA oversight on security breaches," *CCN*, n.d. Accessed: Apr. 19, 2025. [Online]. Available: <https://www.ccn.com/news/crypto/eu-crypto-firms-cyber-audits-esma-security-breaches/>
41. K. Quinlan, "MiCA regulation for node operators and its impact on the digital asset ecosystem," *Rue.ee Blog*, 2023. Accessed: Apr. 19, 2025. [Online]. Available: <https://rue.ee/blog/mica-regulation-for-node/>
42. "Cyber, crypto assets and emerging technology." U.S. Securities and Exchange Commission. Accessed: Apr. 19, 2025. [Online]. Available: <https://www.sec.gov/about/divisions/offices/division-enforcement/cyber-crypto-assets-emerging-technology>
43. T. Orme-Clay, "US markets watchdog sets up new crypto and cybersecurity unit," *Payment Expert*, Feb. 21, 2025. Accessed: Apr. 19, 2025. [Online]. Available: <https://paymentexpert.com/2025/02/21/sec-crypto-and-cyber-security-unit/>
44. V. Capocasale and G. Perboli, "Standardizing smart contracts," *IEEE Access*, vol. 10, pp. 91,203–91,212, 2022, doi: [10.1109/ACCESS.2022.3202550](https://doi.org/10.1109/ACCESS.2022.3202550). Accessed: Apr. 19, 2025. [Online]. Available: <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9869650>
45. "What is a smart contract audit?" Hedera. Accessed: Apr. 19, 2025. [Online]. Available: <https://hedera.com/learning/smart-contracts/smart-contract-audit>
46. "What is the Cardano smart contract certification program." NMKR. Accessed: Apr. 19, 2025. [Online]. Available: <https://www.nmkr.io/glossary/cardano-smart-contract-certification-program>
47. A. Krowne, "Legal challenges in defining and regulating smart contracts," Industria Bus. Lawyers LLP, Miami, FL, USA, Nov. 14, 2024. [Online]. Accessed: Apr. 19, 2025. Available: <https://ibl.law/legal-challenges-in-defining-and-regulating-smart-contracts/>
48. "France's prudential regulator advances work on certification of smart contracts," *Ledger Insights*, Jun. 5, 2024. Accessed: Apr. 19, 2025. [Online]. Available: <https://www.ledgerinsights.com/frances-prudential-regulator-advances-work-on-certification-of-smart-contracts/>
49. M. Orcutt, "States that are passing laws to govern 'smart contracts' have no idea what they're doing," *MIT Technol. Rev.*, Mar. 29, 2018. Accessed: Apr. 19, 2025. [Online]. Available: <https://www.technologyreview.com/2018/03/29/144200/states-that-are-passing-laws-to-govern-smart-contracts-have-no-idea-what-theyre-doing/>
50. "Smart contracts and U.S. state law," Morris, Manning & Martin, LLP, Atlanta, GA, USA, Apr. 5, 2023. Accessed: Apr. 19, 2025. [Online]. Available: <https://www.mmmlaw.com/news-resources/102ibtf-smart-contracts-and-u-s-state-law/>

NIR KSHETRI is a professor of management in the Bryan School of Business and Economics, University of North Carolina at Greensboro, Greensboro, NC 27412 USA. Contact him at nbkshetr@uncg.edu.



CALL FOR SPECIAL ISSUE PROPOSALS

Computer solicits special issue proposals from leaders and experts within a broad range of computing communities. Proposed themes/issues should address important and timely topics that will be of broad interest to *Computer's* readership. Special issues are an essential feature of *Computer*, as they deliver compelling research insights and perspectives on new and established technologies and computing strategies.

Please send us your high-quality proposals for the 2025–2026 editorial calendar. Of particular interest are proposals centered on:

- 3D printing
- Robotics
- LLMs
- AI safety
- Dis/Misinformation
- Legacy software
- Microelectronics

Proposal guidelines are available at:

www.computer.org/csdl/magazine/co/write-for-us/15911





© SHUTTERSTOCK.COM/MICHAEL TRAUTOV

Predicting the Future of Supercomputing

Scott Atchley , Oak Ridge National Laboratory

Rosa M. Badia , Barcelona Supercomputing Center

Bronis R. de Supinski , Lawrence Livermore National Laboratory

Joshua Fryman, Intel

Dieter Kranzlmüller , Leibniz Supercomputing Centre

Srilatha Manne , Advanced Micro Devices, Inc.

Pekka Manninen, IT Center for Science

Satoshi Matsuoka , RIKEN

Dejan Milojicic , Hewlett Packard Labs

Galen Shipman , Los Alamos National Laboratory

Eric Van Hensbergen , Arm

Robert W. Wisniewski , Hewlett Packard Enterprise

The need to solve high-complexity problems using large-scale tightly coupled computing (that is, supercomputing) continues to grow. In this article, we address the needs, challenges, and opportunities for supercomputing over the next decade.

Digital Object Identifier 10.1109/MC.2025.3567232
Date of current version: 27 June 2025

Supercomputing, which involves the use of the highest-performance computing resources available at a given time, has recently seen broader adoption as it is essential for training generative artificial intelligence/machine learning (AI/ML) models. These AI use cases are in addition to the traditional modeling and simulation (modsim) workloads that continue to drive high use at traditional supercomputing centers.

Supercomputing centers are increasingly adopting AI/ML techniques into modsim workloads. This article by leaders from those centers, as well as within the industry, explores the trends and directions that will shape future supercomputers, driven largely by that convergence of modsim and AI/ML techniques. This article extends the predictions of several recent articles that explored the future of supercomputing.^{1,2,3,4,5,6}

EDITOR DEJAN MILOJICICHewlett Packard Labs;
dejan.milovicic@hpe.com

INCREASING USES, INCREASING ADOPTION

As we consider the future of supercomputing, we see several factors that will drive changes to the workloads that are run on supercomputers. These changes will continue to broaden the adoption of supercomputing and will affect the technology used to build supercomputers. In this section and the following one, we describe our expectations for future supercomputing workloads and discuss the technologies that will shape their evolution.

While we expect supercomputing workloads to be augmented with new workloads (for example, AI), we expect that traditional supercomputing workloads will remain a significant use case. These traditional workloads serve a wide range of purposes, from advancing science to deepening our understanding of the universe in which we live, addressing humanity's needs in the modern world, to protecting the national interests of governments that deploy such systems. Nonetheless, we expect these traditional workloads to incorporate new algorithmic techniques, starting with the use of AI/ML models, as has already begun.^{7,8,9} The adoption of AI/ML techniques includes their use to guide the simulated configurations in ensemble calculations but also their use to accelerate expensive calculations of models of physics and biological phenomena.

With the end of Dennard scaling and the slowing of Moore's law, the automatic increase in performance at constant cost and power is over. Modsim practitioners are faced with modest gains in performance with incremental architecture changes. Future gains are largely coming from the increase in silicon within the package. While providing needed performance boosts, it comes with higher power and higher costs for both the additional

silicon and the integration to stitch together several chiplets. When viewed as performance per watt (for example, if a facility has a fixed power budget), then the gains are still modest.

At the same time, the explosive growth in AI, both training and inference, has driven silicon vendors to tailor their products to this lucrative market. It is not clear, however, that modsim can take advantage of lower precision. Some apps will be able to use FP32 for some of their data structures (but not necessarily all) and see benefits compared to lazily promoting everything to FP64. It is not clear if apps will be able to use FP16 for modsim unless it is using AI inferencing in lieu of a component in a multiphysics application, emulation, or iterative refinement. To use ML inferencing, there needs to be an already-trained model. There is a lot of research interest in determining when/if modsim applications can exploit lower precision, which is becoming much more plentiful. There are efforts to see which, if any, apps can use lower precision directly, use lower precision via AI methods, use lower precision via iterative refinement, or use lower precision via emulation. Some apps may be able to do so, while others will not.

The beauty of the General Matrix-Matrix Multiplication (GEMM) emulation methods (that is, Ozaki methods¹⁰) is that precision is finer-grained than with hardware. Hardware is limited to powers of two (for example, FP64, FP32, and FP16), while Ozaki can provide any multiple of four bits (for example, FP40, FP48, and FP56) to provide just enough precision to converge on a valid solution without providing "too much." While Ozaki's scheme can outperform native cuBLAS in some cases, the downsides to emulation are 1) it can only emulate GEMM (that is, matrix-matrix) instructions but not vector instructions,

and 2) it consumes 30–50% of the available memory, thus reducing the solvable problem size. If memory were cheap and plentiful, the latter would not be an issue, but supercomputer users want the fastest memory available. Today, that is high-bandwidth memory, and it is neither cheap nor plentiful. Recently, systems used for AI/ML training have been cast as competitors to supercomputers.

Rather than competitors, the authors view both modsim and AI as having overlapping needs for supercomputer design, except for precision. However, the systems that provide AI/ML capability are best viewed as supercomputers themselves and reflect that AI/ML training has emerged as an important workload for supercomputers. As we look toward the future, not only do we expect that AI/ML training will remain a critical supercomputing workload, but we anticipate that additional new workloads will emerge. We expect that domains that have begun to use supercomputers more extensively due to the success of large-scale AI/ML models, such as finance and retail, will identify new mechanisms to exploit the computational capability available and expand the use of AI/ML in their domain.

The convergence of cloud computing and supercomputing has long been expected. However, this convergence has not fully materialized yet, in part due to the requirements of traditional tightly coupled parallel modsim workloads. Nonetheless, cloud providers continue providing more high performance computing (HPC) capability, and cloud computing continues to be a viable economic and technical alternative for embarrassingly parallel workloads and, as of recently, for AI/ML workloads. They are also suitable for offloading or bursting small-scale experiments and development.

Addressing humanity's needs, such as weather forecasting and biomedical

research, continues to be an important target of supercomputing. These applications include energy needs and its production using nuclear fission near term and fusion long term—but also for fossil fuels and importantly, carbon and water management. Another use is for new materials, particularly for the continued advancement of technology beyond silicon CMOS device scaling. Yet another use case is mitigating and adapting to climate change, including utilizing digital twins.

Digital twins are virtual representations of physical artifacts, systems, or processes with collected real-time information. They enable monitoring, simulation, and prediction of those physical artifacts. Digital twins often use supercomputers directly in a variety of vertical applications and services (for example, for structural analysis, Earth monitoring, manufacturing, and operations) as well as exploit them peripherally (for example, for monitoring, optimizing operation, anomaly detection, or what-if-analysis). Digital twins are used in areas such as the transportation industry, data centers,^{11,12} and even Earth.¹³

Another important use case of traditional supercomputing is helping drive new scientific breakthroughs [that is, helping answer the big questions, for example, performing computation for the follow-on to Laser Interferometer, Gravitational-Wave Observatory (LIGO) or Laser Interferometer Space Antenna (LISA) that will enable sensing of gravitational wavelengths populated by a rich diversity in astrophysical phenomena that are of deep interest to astronomers and astrophysicists]. After a discussion on how the use and adoption of supercomputing evolved, we will next explore how technology evolution impacts workloads.

EVOLVING TECHNOLOGIES AND WORKLOADS

Future supercomputing workloads will reflect recent and anticipated

future technological and industry developments. These trends include not only the adoption of AI/ML to serve edge computing and other end-user applications but also productivity enhancements, such as those driving broad consumer adoption of cloud-based computing. Further, architectural and device-level advancements will continue to motivate new supercomputing application enhancements. This section provides a high-level description of these two influences on future supercomputers. We begin by describing the workloads.

- › New applications are continuing to demand more computational capability, including bioengineering, climate modeling, national security, fusion energy, and many others.
- › HPC and AI will continue to converge and thereby demand more AI-ready infrastructure.
- › Large language models and other models have captured the public imagination, and they open new opportunities in supercomputing.
- › Physics-informed neural networks and other models, possibly integrated into traditional modeling applications, enable the faster exploration of design spaces.
- › Some workloads are increasing performance by leveraging mixed precision computation, while others are leveraging multitenancy to increase performance.
- › Application demand for scale-up networking, including Ultra Accelerator Link (UALink), will continue to increase per-device bandwidth and the number of directly connected scale-up devices, blurring the boundary between scale-up and scale-out infrastructure.

In the last couple of years, advancements in AI, specifically in generative AI applications, have dramatically

influenced private industry toward building large-scale computing infrastructure. Even though these infrastructures are driven by AI requirements, they are becoming increasingly HPC ready. AI and HPC are making significant strides toward convergence, and this development is a major disruptor. We predict the forthcoming technological changes.

- › Accelerators, from traditional (for example, compression and crypto) to ones focused on AI (for example, Cerebras, NextSilicon, and SambaNova) to upcoming (for example, neuromorphic and quantum), will address specialized but important demands, and some are already being incorporated into existing supercomputers. 2.5D and 3D memories present obstacles that must be overcome to use, but they provide significant opportunities to help ameliorate memory wall challenges.
- › Continued evolution of the scale and latency-sensitive industry-standard or standard-compatible/interoperable interconnects (for example, scale-up merging with scale-out) will occur.
- › Increasingly integrated photonics as a means of power reduction, packaging simplicity, and bandwidth enhancement will be seen.
- › Improvements in reliability are driven by the need to address resilience (or fault tolerance) at all levels of the system, from hardware to system software to applications.

These technology changes will result in a new macro-political landscape that may influence decisions on next-generation supercomputer procurement. For example

- › AI will drive technology directions/priorities, including reduced precision, systolics, and fixed function units.

- › Silicon transistor devices are approaching hard limits in scaling, with limited improvements in performance through silicon CMOS scaling, which has implications for specialization, tight integration, and power reduction. These limits introduce a need for deeper co-design alongside other major market forces, such as AI.
- › New computing technologies are being explored, including quantum, neuromorphic, and other accelerators that may substantially change the landscape in terms of scaling, reliability, power, and cooling.
- › Research in new nonvolatile memories (NVMs) has been occurring for many years. If that work leads to successful productization, it may affect the way we design storage, conduct checkpointing, and in general, manage memory.
- › New algorithms (potentially AI inspired and enabled by new accelerators) can also impact performance and scale.

ARCHITECTURE

Two main architectural changes have brought AI and HPC applications closer together. The first is the addition of high-performance GPUs alongside high-performance CPUs for compute, and the second is AI's need for fast and efficient communication within and between compute elements.

One of the biggest shifts in the last decade has been the widespread adoption of GPUs for computation. While accelerated by AI use cases on supercomputers, this trend was occurring independently on HPC systems due to the need for higher compute capabilities while keeping power manageable. Similar motivators (that is, raw performance, performance per watt, performance per area, and performance per dollar) will likely drive the inclusion of accelerator technology (for example, Cerebras, NextSilicon, SambaNova,

and potentially quantum or neuromorphic), though the intercept of the latter two's productive use will require additional time.

As GPUs became dominant, the primary architecture of the system remained homogeneous by node. That is, while each node was heterogeneous (microheterogeneity), the overall system was homogeneous. Many of these new accelerators are not as general purpose as GPUs, and therefore, systems are likely to be macro-heterogeneous. What remains open is the tightness of coupling of these macro-heterogeneous partitions.

The severity of the memory bottleneck in generative AI has led to other forms of acceleration reentering consideration, including computation near memory (CNM) as well as processing in memory (PIM). These computational accelerators, coupled with collective acceleration in the network, data processing units (DPUs), and forms of compute near storage, create a more diverse acceleration landscape than that enabled by GPUs. Further, as chiplet-based design points lead to finer-grained customization, the opportunity to intermingle compute acceleration with general purpose compute may become attractive to better balance system performance, power delivery, and thermal dissipation.

A major block to heterogeneity, whether it be at the micro or macro level, is the programming model. Without a productive programming model that enables efficient offload to accelerators, the additional hardware will not provide a good return on area, cost, or power investment. The transition from CPUs to GPUs was made easier via a programming model and tool stack for GPUs, and any accelerators will have to match those capabilities to be viable. For example, circuits for CNM have been known for more than 50 years,¹⁴ but the general programmability problem remains unsolved and generally avoided as “too hard” to solve.

One of the significant challenges in the post-exascale era is communication.

This challenge involves moving data from memory to compute and between compute. One way to help address this challenge is to move to more tightly coupled architectures. Memory stacking, 2.5D or 3D, has the potential to reduce power and increase bandwidth between compute and memory.

An important aspect of heterogeneous node architectures is moving data between compute elements, specifically between the main CPU and the accelerator. Coarser parallelism leads to less frequent data movement and more efficient use of the accelerator. Traditional HPC applications need serial cores, and many large AI applications are also increasingly benefiting from the utilization of CPUs. Further, many HPC applications remain bulk synchronous with branchy and data-dependent code between parallelizable kernels; that code runs better on CPUs. The AMD Instinct MI300A accelerated processing unit (APU) brings the CPU and the accelerator computing elements together both physically, via chiplets, and programmatically through a unified memory model; Nvidia's Grace-Hopper provides similar benefits using a full reticle CPU and GPU interconnected through NVLink—a chip-to-chip technology. However, hardware and software challenges, such as software offload launch latencies, remain. Tighter coupling may further help improve performance. For example, 3D stacking would allow more memory bandwidth than 2/2.5D integration.

Moving across compute within the same package or same node offers challenges, but significant performance cliffs occur when moving from high-performance nodes to the network due to lower network byte/flop ratios, high network latencies, and high costs of synchronization across nodes. These inefficiencies require application developers to partition their codes in a coarse-grained manner into serial and parallel compute phases, memory movement phases, and network communicator phases with each

one optimized independently. This requirement not only impacts programmer productivity but misses opportunities to optimize power efficiency and memory access across the system. These network inefficiencies also limit strong scaling. The bandwidth and latency cliffs are not the only

latency and bandwidth advantage between these computing elements that were contained within a node. However, the architecture came at a cost. Applications—and particularly communication runtimes—needed to be aware of the topological structure to exploit it.

systems optimized for NVM technology, such as DAOS, VAST, and Weka, are gaining popularity and will increase, including the model stores for AI, such as vector databases. Cloud services have innovated object interfaces, such as S3, that AI frameworks use natively.

The future will determine if ESS leads to a common stack across the community or splinters the community.

inhibitors of performance. The model of how memory is accessed can also have a large and potentially greater impact on the performance of applications when they communicate outside the node. The right internode memory model with enhanced capabilities, such as atomics and load/store access to memory within a supernode, pod, or hypernode (collections of tightly coupled nodes with an enhanced memory model), can improve strong-scaled performance by more than an order of magnitude. Nvidia's NVLink and the UALink standard (which AMD is a part of) are specific solutions that can provide tighter coupling between nodes. The general UALink industry-standard effort is moving to create an interoperable fabric for these needs. Competing pressures on interconnects will likely move future interconnects from low radix high diameter to high radix low diameter to improve efficiencies across a wide spectrum of use cases.

Two decades ago, the connection model was flat. A core comprised a node, and each node had a network connection. The topology varied (for example, butterfly, hypercube, or torus), but all compute elements were uniformly separated. With the introduction of multiple cores per chip, multiple chips within a node, and multiple GPUs within a node, two levels of connectivity, inter- and intranode, were introduced. This architecture provided a communication

Motivated by AI, scale-up networking is creating another layer in the communication hierarchy. Pods, super nodes, wafer scale, or hypernodes, represent an opportunity to connect tens to hundreds (perhaps small thousands) of nodes in a more tightly coupled manner with memory semantics (for example, load/store access and atomic operations). These architectures have better performance for AI and strong-scaled applications but also introduce a programmability cost. Again, the software layers have an opportunity and responsibility to attune the application appropriately for the communication hierarchy.

An open question remains as to the best overall system architecture since this intermediate communication layer (that is, scale-up: between or within a node and across the whole machine) is more expensive from a cost and power perspective than a flat communication architecture. One possibility that shows promise is merging the connectivity emanating from a node into either scale-up or scale-out connectivity. While this approach is a promising notion, no obvious technologies enable it, yet, but the two main standards initiatives in this space, UALink and the Ultra-Ethernet Consortium (UEC), are currently working on it.

Traditionally, the HPC community relied on large-scale hard-drive-based parallel file systems, such as Lustre and GPFS. In recent times, object store file

FACILITIES

Energy has been driving exascale supercomputing as one of the primary constraints. From the beginning of exascale planning, the desire to keep the spending on power to a minimum led to a target of 20 MW.^{15,16} This impacted system designs, specifically cooling, space (the number of racks), and the CPU/GPU ratio. Air cooling was not sufficient, and liquid cooling has become the standard solution for capability-class supercomputers and is seeing broader-based adoption.

Figure 1 notionally presents the evolution of power efficiency on the left (red curve) versus cooling choices on the right (blue curve) during the past few decades. Power efficiency numbers were taken from Oak Ridge National Laboratory supercomputers (Jaguar, Titan, Summit, Frontier). Due to 3D chips, the power density will continue to increase (more than double from 2021 to 2031) according to the IRDS Roadmap,¹⁷ which will require further innovation in cooling, such as immersive or evaporative spray cooling.

In the longer term, both cooling and power requirements may change substantially. Multiple reasons led to the 20-MW limit in the requirements for exascale supercomputers, including cost and the ability to deliver that much power. The new means of energy production, such as small modular reactors (SMRs), are competitively priced per MW and complemented by onsite renewable energy production (for example, wind and solar). If they succeed, they will address both the cost and power delivery to data centers.¹⁸ The AI compute demand and the boom have further shifted the economics and scale of power generation, altering availability and pricing.

SOFTWARE STACK

The system software stack, as defined by everything below an application and above the hardware, continues to increase in complexity. From a modeling and simulation perspective, as the desired capability has increased, system implementers have increasingly turned toward leveraging open source to provide this capability. This change complicates comprehensive testing. The combinations of open source components exponentially increase the number of possible permutations of the software stack. Insufficient connectivity between these open communities (and interest in being connected) has made comprehensive validation significantly more challenging than when a vendor owned all, or most of, the components in a stack.

OpenHPC created a complete and comprehensive system general software stack. Extreme-scale Scientific Software Stack (E4S) of the Exascale Computing Project (ECP) made strides toward unifying the development environment across many open source components. The High Performance Software Foundation (HPSF), unified by Spack, is making strides toward providing optimized software stacks for well-defined systems. Nonetheless, challenges remain, and a stronger community testing effort, perhaps under HPSF, is still needed.

The inclusion of AI software stacks on supercomputers has significantly increased the number of components of the overall software stack. More importantly, AI infrastructure, including the software stack, is undergoing rapid change. The key contributors are investing significant effort to support this rapidly evolving environment while other organizations are challenged to keep up. Overall, the rapid evolution limits the organizations that can stand up and maintain an AI stack, which further increases the need for community efforts toward testing and maintaining the overall software stack.

While E4S was United States centered, Europe is developing the European Software Stack (ESS). The EuroHPC JU will work with stakeholders to coordinate co-design in the research and investigation of hardware and software activities and ensure that those activities meet user requirements and that developed technologies are deployed. Funding is planned for the different building blocks in HPC, AI, and quantum computing (QC) from innovation to deployment, targeting different technical readiness levels as required by the status of hardware developments. Europe will focus on multiple aspects, such as performance and efficiency, AI-software integration, energy consumption,

workflow managers, and support to European processors, among others. The future will determine if ESS leads to a common stack across the community or splinters the community.

As discussed previously, macro-heterogeneity is on the horizon; enhancements of the software will be needed to incorporate the new elements into the system as well as to support macro-heterogeneity generally. To make these accelerators productive, a comprehensive software stack will need to be developed to enable nonexpert application developers. User interfaces, libraries, debuggers, validation tools, high-level programming models, and languages are needed as well as compilers to translate high-level languages to be distributed over coarse-grain reconfigurable architectures or to QC circuits and compilers that adapt already-compiled circuits to a dedicated technology.

As the software stack becomes more complex and the overall user code moves from a single executable to a complex set of interconnected executables, we will need an overarching workflow infrastructure. Some examples of workflow management exist today, but those capabilities will need to be enhanced to cover the great variety of emerging software stacks. They will also require many new capabilities, such as the control of data movement

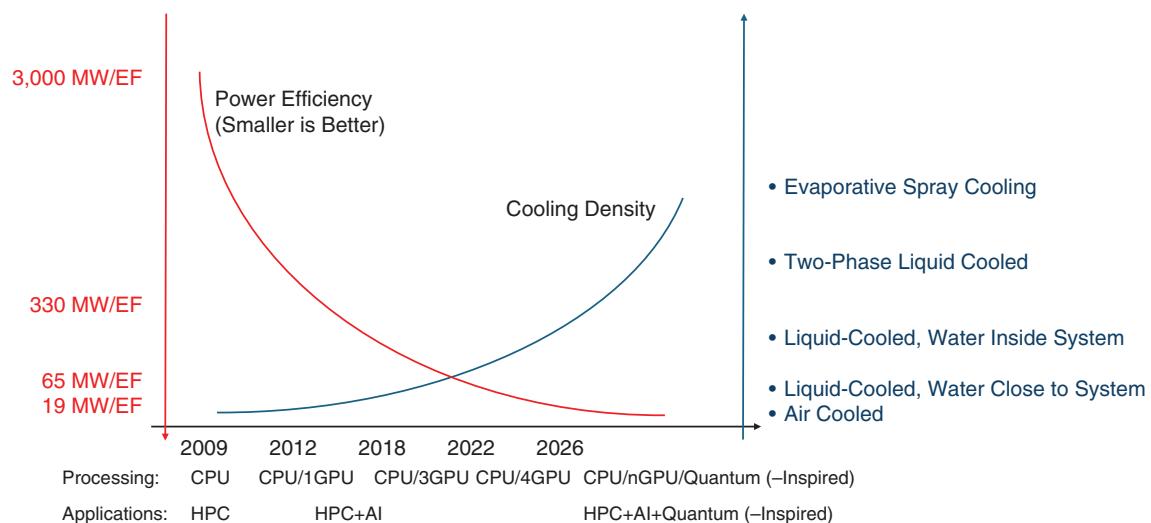


FIGURE 1. Supercomputer power efficiency and cooling over the years.

and enhanced authentication, security, and monitoring.

The amount of power consumed by supercomputers is reaching an inflection point where the cost of electricity throughout the life of the system is approaching its capital cost. New software capabilities must be created to enable users to understand and optimize the tradeoff between performance and energy (for example, to allow a user or system administrator to reduce performance by 10% to save 40% on energy). We will also need support to ramp up and down power more smoothly to meet the requirements of electricity providers.

OPERATIONS

The U.S. ECP was a multibillion-dollar effort, with multiple hundred-million-dollar procurements. In addition, the cost to operate an exascale supercomputer is on the order of 100 million U.S. dollars, a significant part of its total cost of ownership.

Producing and procuring a capability-class supercomputer is a complex operation that is not optimal for the participants in the procurement: regulators, users, integrators, and suppliers. Distributed spending with incremental upgrades could be beneficial. Similarly, the operating expense costs are becoming too high to be financially sustainable. New means of producing and delivering supercomputers could prove beneficial for multiple parties.

Current supercomputers are designed to run applications at an extreme scale. While needed for capability-class applications, this model has challenges for maintenance and partial system refreshes. Accelerator road maps are also more frequent and shorter than the lifetime of supercomputers, which makes refreshes more desirable than in the past, from both the performance and power/cost perspective.

NONFUNCTIONAL REQUIREMENTS

Reliability has long been a focus of traditional HPC, extending from

high-level software to ensure that it did not have any single points of failure, down to the silicon, including both compute and memory. This focus was needed as the high-level fault tolerance model in applications was that if one node failed, the entire application failed. Thus, as the machine grew in node count, it was imperative that reliability was improved. Nevertheless, the mean time between failure on the largest supercomputers has dropped from around a week on emergent petascale systems to a handful of hours on emergent exascale systems. With each generation, new points of hardware and software reliability failures emerge due to ever increasing hardware complexity and software not planning for significant implications of heterogeneous architecture implementations.

Innovations in checkpointing architecture in conjunction with improved bandwidth for checkpoints have predominantly ameliorated the impact that this decreased reliability has on system availability. However, unless something changes, this trend will be unsustainable for the next three orders of magnitude of system performance improvement. Fewer applications can productively employ a full exaflop of compute than the number that could employ a full petaflop. This potentially implies a different usage model for supercomputers in the next decade. Each facility's workload will determine whether petascale or exascale resources (for example, compute, memory capacity, and memory bandwidth) are needed.

AI has only recently been run at large scales. Thus, GPUs have not focused as much on reliability as CPUs that were designed for supercomputers. The AI software stack has also not had years of focus on reliability and ensuring no single point of failure. Recent data from Meta,¹⁹ Alibaba, Google,²⁰ and others show the consequences. As AI continues to scale and systems become larger with the desire to run capability-class applications,

an increased focus on fault tolerance will be needed, both in designing and implementing more reliable hardware and in changing the application fault tolerance model.

AI applications are inherently more resilient to failures because of the nature of their computation. While academic work has explored application-level fault tolerance for modsim applications, it has not been implemented in practice as most of the work could address only specific computational kernels rather than the resilience of the entire application. In one form or another, reliability will need more focus moving forward.

SUMMARY AND OUTLOOK

In this article, we presented our predictions of the future of supercomputing. We first discussed increased use and adoption, followed by evolving technologies and workloads. We then presented the architecture, facilities, software stack, operation, and non-functional requirements. We concluded with some recommendations to critical actors in supercomputing.

Figure 2 and Table 1 summarize our predictions. Figure 2 describes the architecture of future supercomputing, emphasizing the innovations required. Table 1 succinctly presents the evolution of HPC over decades, from traditional to future supercomputing.

Achieving the next level of scale will require innovation, just like it did to get from petascale to exascale. This innovation will likely need to come across the whole system, including new accelerators, interconnects, system software, application and algorithmic innovations, and power and cooling. Some of the scaling may be possible to achieve by leveraging macro-heterogeneity, for example, through the use of AI-specific, quantum or quantum-inspired, or other accelerators in the context of a more traditional GPU-based supercomputer.

Supercomputers will also benefit from the growth in the bandwidth of

interconnects. Photonics could help overcome limited processor shoreline performance, power, and packaging. However, additional investments will have to be made to avoid congestion at scale and to address both jitter and tail latency.

In terms of power and cooling, the current limitations will remain and will have to be addressed with onsite power generation, possibly with SMRs and renewable energy sources as complements to grid supplies. Cooling will require new techniques, as discussed in the “[Facilities](#)” section. Locating data centers in zones where power is cheap and reliable can also help. Areas with abundant water and favorable climates will assist with cooling challenges.

Sustainability is challenging in supercomputing due to the extreme use of power. Some of the approaches of large-scale enterprise data centers can be applied (for example, following the sun or server consolidation)

to a limited extent. Sustainability awareness can help, as can using digital twin techniques to conduct what-if-analyses and understand where opportunities lie.

The use of AI is inherently tied to ethics and is an important topic that will need to be addressed given the widespread use of AI. AI is effective at improving productivity in software development. Productivity in developing supercomputing applications is critical but also hard to automate using AI due to the performance and scale requirements.

RECOMMENDATIONS

We make recommendations to key actors in the supercomputing ecosystem: supercomputer centers, developers, scientists/users, and industry.

Our recommendations for supercomputer centers are as follows:

- Workloads of the future will continue to have demands for

tightly coupled, highly parallel, and noise-free infrastructure at scale. Therefore, the growth in the needed capabilities of future supercomputers will continue, and centers should continue to plan to procure them.

- Future supercomputers may be supplemented by leveraging offload to a public or private cloud or large AI infrastructures for training or services that enhance productivity. Centers should investigate how to incorporate complex workflow capabilities that allow this interaction as well as intrafacility and interfacility workflows. Infrequent delivery of single large supercomputers puts a strain on providers, users, and maintainers of supercomputers. An alternative incremental delivery should be explored to ensure smooth delivery and secure a more reliable

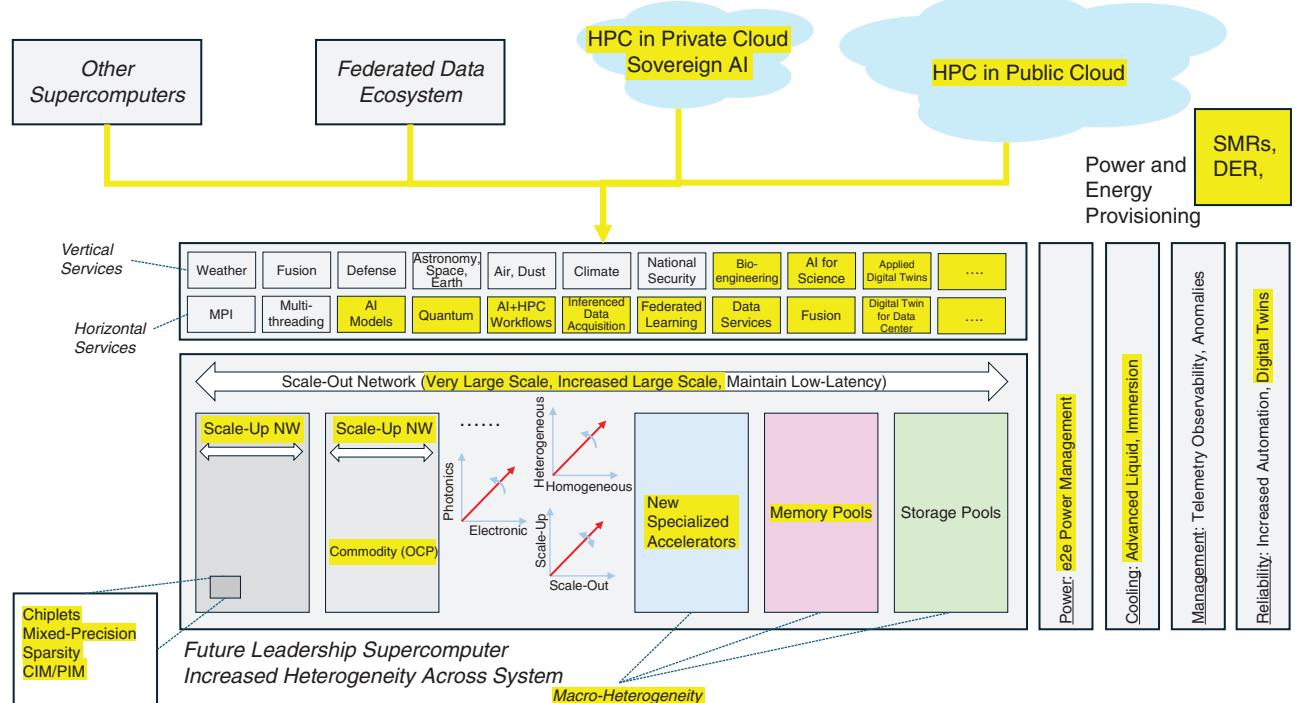


FIGURE 2. High-level supercomputing architecture. Highlighted text in yellow represents new features compared to existing supercomputers. API: application programming interface; SMRs: small modular reactors; DER: distributed energy resources; CIM: computing in memory; OCP: open compute; e2e: end-to-end.

introduction of new features. It also puts HPC at a disadvantage from a performance standpoint. GPU performance is still scaling rapidly, and AI is forcing an acceleration in hardware innovation from compute to networks.

Our recommendations for developers and the open source community are as follows:

- Most of the system software running on supercomputers is becoming open source. The community should become more strategic

about planning and delivering new features and secure approaches and infrastructures to be able to develop and test solutions at scale. To allow the broadest productive use of software, instilling good software engineering practices into community code will be

TABLE 1. Comparing approaches to building and consuming leadership supercomputing systems.

Comparison criteria	Supercomputer eras						
	Traditional HPC supercomputer (1990 to present)*	Grid (2000–2010)	Cloud‡ (2006 to present)	AI cluster training	AI cluster-inference	HPC cloud§	Future HPC supercomputer
How the system is built	Coupling	Very Tight scale-out	Tight scale-out federated	Loose	Tight (scale-up and scale-out)	Loose + scale-up	Loose, scale-up, medium-tight
	Scale	<10× exascale	Multisite (federated?)	Multiregions	Collocated	Distributed	Multiregions
	Reliability	Job-based restarts	Job-based restarts	Cloud like†	Job-based restarts	Cloud like†	Cloud like
	Elasticity	No	Desired	By design	Moderate	Cloud like	Cloud like
	Storage System	Parallel FS (write intensive)	Grid FS	Block, object (read intensive)	Read-training data write-ckpt (file, object)	Read intensive (mostly objects)	Block, object (read intensive)
Consumption model	Business Adoption	Governments	Governments/industry	Consumer/enterprise	Model builders, sovereign AI	Service providers, enterprise	Government/industry/provider
	Networking‡	No (@ Periphery§)	Yes	Inherent	Yes	Yes	Yes
	Multitenancy	Minimal	Yes	Inherent	Moderate (job based)	Yes	Inherent
	Virtualization	No (well, some containers)	Some	Built-in VMs	Containers + K8s	Containers + K8s	Built-in VMs, containers
	Optimized for	Mod/sim HPC	(data-intensive) HPC	Content serving, horizontal scale	Training and tuning large AI models	Models at scale, agents, workflows	Loosely coupled HPC, AI

VMs: virtual machines.

*While the first supercomputer was delivered in 1964, we started counting from 1990 when the first modern scale-out computer was delivered.

†Cloud-like reliability: 1) stateless/fungible VMs; 2) reliable persistence layer (S3, etc.); 3) restartable service requests; and 4) eventual consistency for distributed tasks.

‡Most HPC and AI training is dominantly East-West, while cloud and AI serving are dominantly North-South (N-S). The difference with AI is that it is N-S + scale-up (multi-GPU networks), while the traditional cloud is largely N-S.

§Supercomputers are connected to the outside—but only at the periphery of the system, with a different network.

beneficial (for example, the work E4S did made its components more accessible to a wider community). HPSF is a good step in this direction.

- › As AI is becoming more prevalent in almost every aspect of programming, the models should be treated the same way as open software. The data that were used for training should be made available and documented. While enhancement based on private data will be necessary for some use cases, the data on which open models are based must also be open.
- › In general, but especially for science applications, focus on the explainability of AI methods.
- › Open hardware is becoming an alternative that needs to be carefully evaluated and considered in supercomputing solutions. Open firmware is also an interesting direction to enhance security and maintainability.
- › Work on leveraging low-precision hardware to emulate or perform high-precision calculations is essential. Ultimately, scientific applications need a more rigorous error-based approach to numerical precision.

Our recommendations for scientists and users of supercomputers are as follows:

- › Adjust to using cloud infrastructure and AI programming models combined with the existing traditional HPC algorithms.
- › Continue to be innovative in terms of continuously increased scale and alternative programming models offered by new hardware (for example, AI accelerators and quantum).
- › Invent new algorithms and applications to leverage the new AI and future computing and memory technology.

Our recommendations for industry, integrators, and system vendors are as follows:

- › Ensure sufficient interoperability across the components and interconnects to enable reusability across supercomputers.
- › Provide sufficient documentation and interfaces for using hardware and core system software.
- › Support interfaces and software for the maintenance and management of supercomputers at scale.
- › Provide the capability to combine AI capability productively into existing applications.

The need for supercomputing continues to grow. In addition to the needs of traditional scientific computing, AI's needs are driving the evolution of computing hardware and software. The authors lay out several challenges and opportunities for the next decade for computing facilities; developers, scientists and users; and industry. [C]

ACKNOWLEDGMENT

We would like to thank Cullen Bash, Paolo Faraboschi, and Samantika Sury for their advice and insights that substantially helped improve this article. This manuscript has been authored in part by Lawrence Livermore National Security, LLC under Contract DE-AC52-07NA2 7344 with the U.S. Department of Energy (DOE) (Grant LLNL-JRNL-2005193); it is based in part upon work supported by the DOE, National Nuclear Security Administration under Contract 89233218CNA000001; and it used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725. LA-UR-25-23959 (approved for public release). Rosa M. Badia acknowledges funding from Projects CEX2021-001148-S, and PID2023-147979NB-C21 from the MCIN/AEI and MICIU/AEI/10.13039/

501100011033 and by FEDER, UE, by the Departament de Recerca i Universitats de la Generalitat de Catalunya, research group MPiEDist (2021 SGR 00412). AMD, the AMD Arrow logo, AMD CDNA, AMD Instinct, AMD ROCm, AMD Infinity Cache, AMD Infinity Fabric, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

REFERENCES

1. R. M. Badia, I. Foster, and D. Milojicic, "Future of HPC," *IEEE Internet Comput.*, vol. 27, no. 1, pp. 5–6, Jan./Feb. 2023, doi: [10.1109/MIC.2022.3228323](https://doi.org/10.1109/MIC.2022.3228323).
2. D. Milojicic, P. Faraboschi, N. Dube, and D. Roweth, "Future of HPC: Diversifying heterogeneity," in *Proc. Design, Automat. Test Europe Conf. Exhib. (DATE)*, 2021, pp. 276–281, doi: [10.23919/DATExpo51398.2021.9474063](https://doi.org/10.23919/DATExpo51398.2021.9474063).
3. N. Dube, D. Roweth, P. Faraboschi, and D. Milojicic, "Future of HPC: The internet of workflows," *IEEE Internet Comput.*, vol. 25, no. 5, pp. 26–34, Sep./Oct. 2021, doi: [10.1109/MIC.2021.3103236](https://doi.org/10.1109/MIC.2021.3103236).
4. G. M. Shipman et al., "The future of HPC in nuclear security," *IEEE Internet Comput.*, vol. 27, no. 1, pp. 16–23, Jan./Feb. 2023, doi: [10.1109/MIC.2022.3229037](https://doi.org/10.1109/MIC.2022.3229037).
5. E. Deelman et al., "High-performance computing at a crossroads," *Science*, vol. 387, no. 6736, pp. 829–831, 2025, doi: [10.1126/science.adu0801](https://doi.org/10.1126/science.adu0801).
6. R. Stevens, V. Taylor, J. Nichols, A. B. Maccabe, K. Yellick, and D. Brown, "AI for science: Report on the Department of Energy (DOE) Town Halls on artificial intelligence (AI) for science," Argonne National Lab. (ANL), Argonne, IL, USA, Tech. Rep. ANL-20/17; 158802; TRN: US2103893, Feb. 2020.
7. W. Jia et al., "Pushing the limit of molecular dynamics with ab initio accuracy to 100 million atoms with machine learning," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Atlanta, GA, USA, 2020, pp. 1–14.

8. S. Das et al., "Large-scale materials modeling at quantum accuracy: Ab initio simulations of quasicrystals and interacting extended defects in metallic alloys," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Denver, CO, USA, 2023, pp. 1-12.
9. G. Dharuman et al., "MProt-DPO: Breaking the ExaFLOPS barrier for multimodal protein design workflows with direct preference optimization," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Atlanta, GA, USA, 2024, pp. 1-13.
10. H. Ootomo, K. Ozaki, and R. Yokota, "DGEMM on integer matrix multiplication unit," 2024, *arXiv:2306.11975*.
11. J. Athavale et al., "Digital twins for data centers," *Computer*, vol. 57, no. 10, pp. 151-158, Oct. 2024, doi: [10.1109/MC.2024.3436945](#).
12. W. Brewer et al., "A digital twin framework for liquid-cooled supercomputers as demonstrated at exascale," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Atlanta, GA, USA, 2024, pp. 1-18.
13. L. R. Leung, "Earth system modeling or actionable science," *Pacific Northwest Nat. Lab.*, Richland, WA, USA, Jul. 2024. [Online]. Available: https://www.nersc.gov/assets/Uploads/NERSC_Leung_2024_final.pdf
14. R. R. Seeber, "Associative self-sorting memory," presented at the *Eastern Joint IRE-AIEE-ACM Comput. Conf. (IRE-AIEE-ACM)*, New York, NY, USA: ACM, Dec. 13-15, 1960, pp. 179-187.
15. P. Kogge et al., "ExaScale computing study: Technology challenges in achieving exascale systems," DARPA, Arlington, VI, USA, Sep. 2008. [Online]. Available: https://ftp.eecs.berkeley.edu/~yelick/papers/Exascale_final_report.pdf
16. S. Atchley et al., "Frontier: Exploring exascale the system architecture of the first exascale supercomputer," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Denver, CO, USA, 2023, pp. 1-16, doi: [10.1145/3581784.3607089](#).
17. "International Roadmap for Devices and Systems™ 2023 update: Systems and architectures," IEEE, Piscataway, NJ, USA, 2023. [Online]. Available: https://irds.ieee.org/images/files/pdf/2023/2023IRDS_Perspectives.pdf
18. C. Bash, J. Bian, D. Milojicic, C. D. Patel, L. Strezoski, and V. Terzija, "Energy supplies for future data centers," *Computer*, vol. 57, no. 7, pp. 126-134, Jul. 2024, doi: [10.1109/MC.2024.3393248](#).
19. A. Grattafiori et al., "The Llama 3 herd of models," 2024, *arXiv:2407.21783*.
20. N. Jouppi et al., "TPU v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings," in *Proc. 50th Annu. Int. Symp. Comput. Archit. (ISCA)*, New York, NY, USA: ACM, 2023, pp. 1-14.
21. C. Kesselman and I. Foster, *The Grid: Blueprint for a New Computing Infrastructure*. Burlington, MA, USA: Morgan Kaufmann, 1999.
22. A. Gupta et al., "Evaluating and improving the performance and scheduling of HPC applications in cloud," *IEEE Trans. Cloud Comput.*, vol. 4, no. 3, pp. 307-321, Jul./Sep. 2016, doi: [10.1109/TCC.2014.2339858](#).
23. T. Gamblin et al., "HPC center of the future: R&D acquisition intent," Lawrence Livermore Nat. Lab., Livermore, CA, USA, Tech. Rep. LLNL-TR-871269, Nov. 14, 2014.

SCOTT ATCHLEY is the CTO at Oak Ridge National Laboratory's National Center for Computational Science, Oak Ridge, TN 37830 USA. Contact him at scott@ornl.gov.

ROSA M. BADIA is a workflow and distributed computing manager at the Barcelona Supercomputing Center, 08034 Barcellona, Spain. Contact her at rosa.m.badia@bsc.es.

BRONIS R. DE SUPINSKI is the CTO for Livermore Computing at the Lawrence Livermore National Laboratory, Livermore, CA 94550 USA. Contact him at bronis@llnl.gov.

JOSHUA FRYMAN is a fellow at Intel, Hillsboro, OR 97124 USA. Contact him at joshua.b.fryman@intel.com.

DIETER KRANZLMÜLLER is the chair of the board of directors at the Leibniz Supercomputing Centre (LRZ), 85748 Garching bei München, Germany. Contact him at dieter.kranzlmueller@lrz.de.

SRLATHA MANNE is a senior fellow at Advanced Micro Devices, Inc., Seattle, WA 98103 USA. Contact her at srilatha.manne@amd.com.

PEKKA MANNINEN is the director of science and technology at CSC, the Finnish IT Center for Science, 02101 Espoo, Finland. Contact him at pekka.manninen@csc.fi.

SATOSHI MATSUOKA is the director of THE RIKEN Center for Computational Science, Saitama 351-01, Japan. Contact him at matsu@acm.org.

DEJAN MILOJICIC is an HPE fellow and vice president at Hewlett Packard Labs, Milpitas, CA 95035 USA. Contact him at dejan.milojicic@hpe.com.

GALEN SHIPMAN is a computer scientist at the Los Alamos National Laboratory, Los Alamos, NM 87545 USA. Contact him at gshipman@lanl.gov.

ERIC VAN HENSBERGEN is a fellow at ARM, Austin, TX 78735 USA. Contact him at eric.vanhensbergen@arm.com.

ROBERT W. WISNIEWSKI is an HPE fellow, chief architect, and vice president of AI and HPC Solutions at Hewlett Packard Enterprise, Spring TX 77389 USA. Contact him at robert.wisniewski@hpe.com.



IEEE Computer Society Volunteer Service Awards

Nominations accepted throughout the year.

T.Michael Elliott Distinguished Service Certificate

Highest service award in recognition for distinguished service to the IEEE Computer Society at a level of dedication rarely demonstrated. i.e., initiating a Society program or conference, continuing officership, or long-term and active service on Society committees.

Meritorious Service Certificate

Second highest level service certificate for meritorious service to an IEEE Computer Society-sponsored activity. i.e., significant as an editorship, committee, Computer Society officer, or conference general or program chair.

Outstanding Contribution Certificate

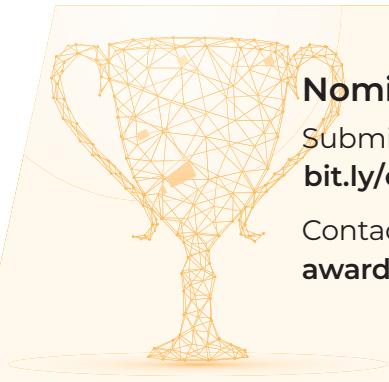
Third highest level service certificate for a specific achievement of major value to the IEEE Computer Society, i.e., launching a major conference series, a specific publication, standards and model curricula.

Continuous Service Certificate

Recognize and encourage ongoing involvement of volunteers in IEEE Computer Society programs. The initial certificate may be awarded after three years of continuous service.

Certificate of Appreciation

Areas of contribution would include service with a conference organizing or program committee. May be given to subcommittee members in lieu of a letter of appreciation.



Nominations

Submit your nomination at
bit.ly/computersocietyawards

Contact us at
awards@computer.org





Can AI Fix Buggy Code? Exploring the Use of Large Language Models in Automated Program Repair

Lan Zhang^{ID}, Northern Arizona University

Anoop Singh^{ID}, National Institute of Standards and Technology

Qingtian Zou, University of Texas Southwestern Medical Center

Xiaoyan Sun^{ID}, Worcester Polytechnic Institute

Peng Liu^{ID}, The Pennsylvania State University

This article reviews the current human–large language models collaboration approach to bug fixing and points out the research directions toward (the development of) autonomous program repair artificial intelligence agents.

The field of software engineering has witnessed a paradigm shift with the advent of large language models (LLMs). These sophisticated artificial intelligence (AI) systems have demonstrated remarkable versatility across various software development tasks, including code generation, bug detection, and code review.^{1,2,3} The potential of LLMs

Digital Object Identifier 10.1109/MC.2025.3527407
Date of current version: 27 June 2025



to revolutionize software development practices has sparked broad interest within both academic and industry circles, prompting a surge of research into their capabilities and limitations.

A recent breakthrough in this domain came with the introduction of Devin, an LLM-powered AI system capable of autonomously completing 13.8% of real-world coding tasks.⁴ These tasks encompass a range of complex operations, from diagnosing and fixing bugs to conducting comprehensive code reviews. However, the relatively modest success rate of 13.8% in real-world scenarios raises a critical question that forms the core of our investigation: Are we truly prepared to leverage LLMs for repairing buggy complex programs? This question is not merely academic but has far-reaching implications for the future of software development and maintenance practices.

To address this fundamental quest, our study focuses on two modes of LLM-supported program repair:

- › **Human–LLM collaboration:** This approach examines the synergistic relationship between human software engineers and LLMs in the bug repair process.⁵ It encompasses both interactive, dialogue-based methodologies and more integrated solutions, such as real-time code completion and suggestion systems.
- › **Autonomous AI agent repair:** This mode investigates the potential for LLMs to independently identify and rectify bugs without direct human intervention, representing a more ambitious vision of automated program repair.

By examining the efficacy of LLMs across diverse programming contexts—for example, C/C++, Java, and Python—we aim to provide a nuanced

understanding of their current capabilities and limitations in addressing complex software bugs. Our findings reveal a nuanced landscape of LLM-supported program repair. For the human–LLM collaboration mode, we observed that results could be significantly improved when humans provide additional contextual knowledge. This includes information about variable contexts, relevant data structures, related functions, and even the underlying logic of the code. This synergy between human expertise and LLM capabilities shows promise for enhancing bug repair processes in complex software systems. In contrast, the autonomous AI agent repair mode presents a more challenging frontier. Our research indicates that we are still far from achieving reliable automatic code repair using LLMs alone. The complexity of real-world software systems, coupled with the nuanced understanding required for effective bug repair, continues to pose significant challenges for fully autonomous LLM-based solutions.

HUMAN–LLM COLLABORATION

GitHub Copilot's ROBIN system represents a significant advancement in human–LLM collaboration for debugging.⁶ It uses multiple AI agents to analyze code context, exception information, and user queries, guiding developers through systematic debugging steps. ROBIN leverages LLMs as reasoning engines to provide interactive and collaborative debugging assistance through a chat-based interface. It analyzes exception information, code context, and user queries, guiding developers through a series of steps to explore potential hypotheses, gather more information, and utilize IDE debugging tools to fix issues. This industrial work demonstrates the potential for more effective collaboration

DISCLAIMER

Commercial products are identified to adequately specify certain procedures. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the identified products are necessarily the best available for the purpose.

between developers and AI in software debugging tasks.

To understand the current state and potential of human–LLM collaboration in program repair, we conducted a comprehensive review of existing research across multiple programming languages and methodologies. Table 1 summarizes our findings, categorizing studies based on programming language (C/C++, Java, Python), dataset type (synthetic programs and real-world projects), and methodology.

Our analysis reveals a clear trend across all three programming languages: The performance of LLM-assisted repair techniques tends to decrease as the complexity of the dataset increases.^{2,7,8,9,15} For example, Yang et al.⁷ and Pearce et al.⁸ investigated human-supported dialogue approaches with synthetic programs, achieving remarkably high success rates: up to 100% in some cases. However, when this methodology was extended to real-world projects by Zhang et al.,² Kulsum et al.,⁹ and again by Pearce et al.,⁸ the performance dropped dramatically to less than 20%.

While the general trends are consistent across C/C++, Java, and Python, some language-specific nuances emerged. For instance, as depicted in “Example 1: Syntax Difference in Java and C++,” Java manages references to objects without explicit pointers, while C++ allows direct memory

TABLE 1. LLM-based program repair across languages and methodologies.

	Dataset type	Methodologies	Article
C/C++	Synthetic programs	Human-supported dialogue	Yang et al., ⁷ Pearce et al. ⁸
	Real-world projects	Human-supported dialogue	Zhang et al., ² Bajpai et al., ⁶ Pearce et al., ⁸ Kulsum et al. ⁹
Java	Synthetic programs	Human and static tools supported dialogue	Kang et al., ¹⁰ Xia and Zhang, ¹¹ Wadhwa et al. ¹²
	Real-world projects	Human-supported dialogue	Kulsum et al. ⁹
Python	Synthetic programs	Human and static tools supported dialogue	Wadhwa et al., ¹² Lemieux et al., ¹³ Cao et al. ¹⁴
	Real-world projects	Human and static tools supported dialogue	Parasaram et al., ¹⁵ Jimenez et al. ¹⁶

EXAMPLE 1: SYNTAX DIFFERENCE IN JAVA AND C++.

Java:

```
String str = "Hello";
List<Integer> numbers = new
ArrayList<>();
System.out.println(str.length());
```

C++:

```
std::string str = "Hello";
int* numbers = new int(5);
std::cout << (*numbers) << std::endl;
delete numbers;
```

manipulation through pointers ("str"). Moreover, Java employs automatic memory management through garbage collection, where "numbers" is automatically deallocated when it's no longer referenced or goes out of scope. In C++, we must manually allocate memory with "new" and then explicitly deallocate it with "delete" to prevent memory leaks. Similar to Java, Python uses automatic memory management. Its dynamic typing and high-level abstractions can simplify certain programming tasks, potentially making some types of repairs more straightforward. For example, Python shows the highest success rate at 38.80%,¹⁶ while C/C++ lags behind at 16.5%.² The lower performance in

C/C++ can be partially attributed to the complexity introduced by manual memory management and pointer manipulation.

Our analysis of the human-LLM collaboration in program repair leads to one key conclusion: Human expertise continues to play a critical role in the bug repair process. Results improve substantially when humans provide additional contextual knowledge.^{2,5}

- **Context of variables:** Understanding the context of variables is crucial for LLMs in program repair for several reasons. The scope of a variable, whether it's global, local, or class-level, determines where it can be accessed and modified. LLMs need to understand this to avoid introducing bugs by incorrectly accessing or modifying variables. Knowing the range of possible values a variable can take helps in identifying potential edge cases or unexpected inputs that could lead to vulnerabilities. Understanding how a variable is typically used within the code, such as a loop counter, a flag, or to store intermediate results, helps LLMs generate more appropriate and context-aware fixes. Tracking how the value of a variable changes throughout the program's execution is essential for identifying the root cause of bugs and proposing effective solutions. In dynamically typed languages, inferring the type of a variable

from its usage context is crucial for generating type-safe patches.

- **External elements:** Knowledge of external functions, data structures, and variables is vital for LLMs in program repair. LLMs need to understand the correct usage of external application programming interfaces, including function signatures, return values, and potential side effects. For languages with manual memory management, understanding how external functions allocate and deallocate memory is crucial for preventing memory leaks and buffer overflows. Knowledge of how external functions report errors, such as through return codes or exceptions, is necessary for implementing proper error checking and handling in patches. Understanding whether external functions are thread-safe is important when generating patches for multithreaded applications.

- **Logic of the vulnerable code:** Comprehending the logic of vulnerable code is essential for effective program repair. Understanding what the code is supposed to do is crucial for ensuring that patches fix the vulnerability without breaking the intended functionality. Analyzing the control flow of the vulnerable code helps in identifying potential logical errors or improper handling of edge cases. Understanding how different parts of the code

CASE STUDY 1: UNDERSTANDING THE RANGE OF A VARIABLE.

I will provide a code that has bug: CWE-119...(shorten to save space)

```
1 static void tokenadd(struct jv_parser *p, char c) {  
2     assert(p->tokenpos <= p->tokenlen);  
3     /**<start-bug>**/  
4     if (p->tokenpos == p->tokenlen) {  
5         /**<end-bug>**/  
6         p->tokenlen = p->tokenlen * 2 + 256;  
7         p->tokenbuf = jv_mem_realloc(p->tokenbuf, p->tokenlen);  
8     }  
9     assert(p->tokenpos < p->tokenlen);  
10    p->tokenbuf[p->tokenpos++] = c;  
11 }
```

Reply from ChatGPT (shorten to save space): The issue here is that there's a potential to write outside of the allocated memory for `tokenbuf` in the last line of code.

```
1 ...  
2 /**<start-bug>**/  
3 if(p->tokenpos >= p->tokenlen){  
4     p->tokenlen = p->tokenlen*2 + 256;  
5     char* newbuf = jv_mem_realloc(p->tokenbuf, p->tokenlen);  
6     if(newbuf == NULL) {  
7         fprintf(stderr, "Memory allocation failed\n");  
8         return;  
9     }  
10    p->tokenbuf = newbuf;  
11 }  
12 /**<end-bug>**/  
13 assert(p->tokenpos<p->tokenlen);  
14 ...
```

Additional knowledge: Off-by-one error in the `tokenadd` function allows remote attackers to cause a denial of service (crash), which triggers a heap-based buffer overflow. If `p->tokenpos` is more than or equal to the end of buffer, resize the buffer. The end of buffer should be `p->tokenlen - 1` not `p->tokenlen`.

Reply from ChatGPT (shorten to save space):

```
1 ...  
2 /**<start-bug>**/  
3 if(p->tokenpos>=p->tokenlen-1){  
4     p->tokenlen = p->tokenlen*2 + 256;  
5     char* newbuf = jv_mem_realloc(p->tokenbuf, p->tokenlen);  
6     if(newbuf == NULL) {  
7         fprintf(stderr, "Memory allocation failed\n");  
8         return;  
9     }  
10    p->tokenbuf = newbuf;  
11 }  
12 /**<end-bug>**/  
13 ...
```

depend on each other's outputs helps in ensuring that patches maintain the correct data flow. Grasping the underlying algorithms used in the code is necessary for proposing optimizations or alternative implementations that might resolve the vulnerability. Identifying implicit security assumptions in the code, such as trust boundaries and input validation, is important for generating patches that enhance the overall security posture.

In “Case Study 1: Understanding the Range of a Variable,” providing comprehensive information about the vulnerable code’s logic and the relationship between the variables `p-> tokenpos` and `p-> tokenlen` greatly helps LLMs generate effective patches. This additional knowledge should involve explaining the fundamental logic behind the vulnerability, emphasizing the potential disparity between `p-> tokenpos` and `p-> tokenlen`, and clarifying the proper boundary conditions. Specifically, elucidating that `p-> tokenbuf` is capable of holding `p-> tokenlen` elements, making the maximum index `p-> tokenpos` should be `p-> tokenlen-1`.

AI AGENT AUTOMATIC REPAIR

Recent advancements in AI have led to the development of increasingly sophisticated coding assistants, such as Codeium,¹⁷ Devin,⁴ Cursor,¹⁸ Magic,¹⁹ Replit,²⁰ and Cody.²¹ Devin AI represents a significant advancement in AI-powered software engineering, demonstrating impressive capabilities in autonomously resolving GitHub issues. This system can build and deploy applications end-to-end, encompassing tasks, such as project cloning, exploration of structures related to vulnerable functions, test case updating based on compiler error messages, generation of new test cases through brute force methods, and bug identification and repair. Devin’s ability to resolve 13.8% of issues in the SWE-bench benchmark,²²

outperforming GPT-4 by a factor of three, is a notable technical achievement. While tools like GitHub Copilot, Codeium, and Cody primarily focus on code completion and generation, their underlying technologies contribute to the broader field of automatic code repair. These systems leverage LLMs trained on vast corpora of code, enabling them to understand code context and suggest fixes for common errors. However, it is crucial to contextualize this success within the broader landscape of software development. While capabilities of automatic AI agents are impressive, the full realization of automatic repair in practical, large-scale software development environments remains a challenging goal that will require further advancements in AI technology and software engineering practices.

FULLY AUTOMATIC AI AGENT ARE IN EARLY STAGE

Despite the promising advancements, automatic code repair using AI face several significant challenges.

- › **Program comprehension:** While Devin has demonstrated enhanced code context understanding compared to previous systems, it still encounters difficulties when faced with complex projects involving numerous interdependent components. For instance, in the case of scikit-learn-11542,²³ Devin identified only two instances with inconsistent default values. However, a comprehensive analysis starting from the RandomForestClassifier would reveal five such instances. This discrepancy highlights the need for more sophisticated algorithms capable of traversing and understanding complex dependency graphs in large-scale software projects.
- › **Verification and testing:** Devin's current approach for verification and testing relies heavily on human-written test cases and brute force input generation techniques. While this methodology

can be effective for simple problems, it falls short when dealing with the complexities of real-world software systems. For example, in scikit-learn-25744,²⁴ Devin correctly identified that the issue stemmed from the min_samples_split parameter and implemented an error message for integer values less than 2. However, it failed to verify the error condition for float values of min_samples_split, which should be constrained between 0.0 and 1.0 (exclusive) when representing a percentage. This oversight underscores the potential dangers of incomplete verification, particularly in critical systems where such oversights could lead to severe consequences.

- › **Contextual understanding:** Automatic repair systems must not only fix the immediate bug but also ensure that the repair aligns with the broader context of the software, including design patterns, coding standards, and project-specific requirements. This level of contextual understanding remains a significant hurdle for current AI systems.

FUTURE DIRECTIONS IN AUTOMATIC CODE REPAIR

Recent research in AI-driven program repair has shown promising results, particularly in addressing well-defined programming tasks of limited scope.^{25,26,27} These works have made significant strides by letting AI agents leverage static and dynamic analysis tools to examine compilation information and code output. This integrated approach guides AI agents in their repair efforts, improving the accuracy and reliability of the generated fixes. However, ensuring the correctness and reliability of AI-generated repairs remains a critical challenge, particularly as we move toward more complex systems. The field of AI-driven program repair continues to evolve, with several promising areas for future research.

Advanced program understanding

While recent models have improved in understanding code context, they still struggle with grasping the full scope of a program, including external dependencies, project-specific conventions, and broader architectural considerations. Developing more sophisticated techniques to capture semantic information and programmer intent is crucial for the future of AI-driven program repair. This may involve leveraging advanced natural language processing techniques to better interpret code comments and documentation.^{28,29} Incorporating program dependency analysis could enhance the AI's understanding of the context and potential impact of repairs.² Additionally, utilizing machine-learning models trained on vast codebases could help in recognizing common patterns and idioms in software design.^{4,6}

Rigorous verification and testing

One of the most significant challenges is ensuring the correctness of AI-generated patches. While AI models can generate plausible fixes, they may introduce new bugs or fail to fully address the underlying issue. Developing robust verification mechanisms for AI-generated patches remains an open problem. This involves integrating formal verification techniques with AI-generated repairs to provide mathematical guarantees of correctness.²⁵ Developing specialized testing frameworks that can automatically generate comprehensive test suites for AI-repaired code would help ensure the reliability of the fixes.³⁰ Additionally, utilizing symbolic execution and model checking techniques would allow for systematic exploration of the state space of repaired programs.²⁹

Multilevel software reasoning

Enhancing AI models' ability to reason about software at various levels of abstraction is essential for comprehensive program repair. For example, GPT-01 can reason through complex tasks and solve harder problems than

previous models in science, coding, and math.³¹ Future work could focus on developing hierarchical models that can simultaneously consider low-level code logic and high-level software system architectures.³² Exploring reinforcement learning approaches might allow AI agents to learn from the consequences of their repair decisions across different abstraction levels.^{33,34,35} By improving the AI's ability to reason at multiple levels, we can expect more sophisticated repairs that consider both local code improvements and their global impact on the system.

Explainability and transparency

From the perspective of AI agents, even as they are expected to work autonomously, the role of human supervision remains crucial, especially in the development and maintenance of critical systems. This underscores the importance of explainability and transparency in AI-driven program repair. Motivated by the need to bridge the gap between AI capabilities and human oversight, future work in this area should focus on several key aspects. Shapley additive explanation values could quantify the importance of different code features (for example, specific lines, functions, or dependencies) in the AI's decision to make a particular repair. This would allow human supervisors to understand which parts of the code most influenced the AI's choice of repair strategy.³⁶ Developing sophisticated attention mechanisms could highlight specific parts of the code that influence the AI's repair decisions, providing insight into the agent's focus and reasoning process.³⁷

By addressing these key areas, researchers aim to bridge the gap between current capabilities and the vision of fully autonomous AI agents capable of general-purpose program repair. While this goal remains distant, ongoing advancements in these areas continue to push the boundaries of what's possible in AI-driven software development and bug fixing.

Our investigation into the capabilities of LLMs in program repair reveals a nuanced landscape with significant implications for software engineering. In the realm of human-LLM collaboration, our findings demonstrate a promising synergy, where human expertise in providing contextual knowledge significantly enhances LLMs' effectiveness in bug repair processes. This collaborative approach shows great potential for improving software development and bug fixing practices, particularly in complex systems. However, the results for autonomous AI agent repair indicate that we are still far from achieving reliable, fully autonomous code repair using LLMs alone. These findings lead us to conclude that while LLMs represent a powerful tool in software engineering, they are not yet ready to replace human expertise in program repair. The most promising path forward appears to be a hybrid approach that leverages the strengths of both human developers and LLMs. As we move forward, it is crucial to focus on enhancing LLMs' contextual understanding, developing more sophisticated human-LLM interfaces, and improving LLMs' ability to reason about and verify their proposed solutions. By maintaining a balanced perspective and working toward solutions that harmoniously combine human expertise and AI, we can continue to advance the field of software development. □

ACKNOWLEDGMENT

Peng Liu was supported by National Science Foundation (NSF) CNS-2019340, NSF ECCS-2140175, and National Institute of Standards and Technology 60NANB22D144. Xiaoyan Sun is supported by NSF DGE2105801.

REFERENCES

1. Z. Wang, L. Zhang, and P. Liu, "ChatGPT for software security: Exploring the strengths and limitations of ChatGPT in the security applications," 2023, arXiv:2307.12488.
2. L. Zhang, Q. Zou, A. Singhal, X. Sun, and P. Liu, "Evaluating large language models for real-world vulnerability repair in C/C++ code," in Proc. 10th ACM Int. Workshop Secur. Privacy Analytics, New York, NY, USA, 2024, pp. 49–58, doi: 10.1145/3643651.3659892.
3. H. Joshi, J. C. Sanchez, S. Gulwani, V. Le, I. Radiček, and G. Verbruggen, "Repair is nearly generation: Multi-lingual program repair with LLMs," in Proc. 37th AAAI Conf. Artif. Intell. (AAAI), 2023, vol. 37, no. 4, pp. 5131–5140, doi: 10.1609/aaai.v37i4.25642.
4. S. Rozendal. "Breakthrough in AI agents?" Effective Altruism Forum. Accessed: Apr. 10, 2024. [Online]. Available: <https://forum.effectivealtruism.org/posts/3brE2Mt6qC-72cQvzL/breakthrough-in-ai-agents-on-devin-the-zvi-linkpost>
5. A. Sellen, and E. Horvitz. "The rise of the AI co-pilot: Lessons for design from aviation and beyond," 2023, arXiv:2311.14713.
6. Y. Bajpai et al., "Let's fix this together: Conversational debugging with GitHub Copilot," in Proc. IEEE Symp. Vis. Lang. Human-Centric Comput. (VL/HCC), 2024, pp. 1–12.
7. B. Yang et al., "CREF: An LLM-based conversational software repair framework for programming tutors," 2024, arXiv:2406.13972.
8. H. Pearce, B. Tan, B. Ahmad, R. Karri, and B. Dolan-Gavitt, "Examining zero-shot vulnerability repair with large language models," in Proc. IEEE Symp. Secur. Privacy (SP), 2023, pp. 2339–2356, doi: 10.1109/SP46215.2023.10179420.
9. U. Kulsum, H. Zhu, B. Xu, and M. d'Amorim, "A case study of LLM for automated vulnerability repair: Assessing impact of reasoning and patch validation feedback," in Proc. 1st ACM Int. Conf. AI-Powered Softw., New York, NY, USA, 2024, pp. 103–111, doi: 10.1145/3664646.3664770.
10. S. Kang, J. Yoon, and S. Yoo, "Large language models are few-shot testers: Exploring LLM-based general bug reproduction," 2022, arXiv:2209.11515.
11. C. S. Xia, and L. Zhang, "Keep the conversation going: Fixing 162 out of 337 bugs for \$0.42 each using ChatGPT," 2023, arXiv:2304.00385.

12. N. Wadhwa et al., "Frustrated with code quality issues? LLMs can Help!" 2023, *arXiv:2309.12938*.
13. C. Lemieux, J. Priya Inala, S. K. Lahiri, and S. Sen, "CodaMosa: Escaping coverage plateaus in test generation with pre-trained large language models," in Proc. IEEE/ACM 45th Int. Conf. Softw. Eng. (ICSE), Melbourne, Australia, 2023, pp. 919–931, doi: [10.1109/ICSE48619.2023.00085](https://doi.org/10.1109/ICSE48619.2023.00085).
14. J. Cao, M. Li, M. Wen, and S-c Cheung, "A study on prompt design, advantages and limitations of ChatGPT for deep learning program repair," 2023, *arXiv:2304.08191*.
15. N. Parasharam et al., "The fact selection problem in LLM-based program repair," 2024, *arXiv:2404.05520*.
16. C. E. Jimenez et al., "SWE-bench: Can language models resolve real-world GitHub issues?" in Proc. 12th Int. Conf. Learn. Representations, 2024.
17. "Windsurf editor and Codeium extensions." Codeium. [Online]. Available: <https://codeium.com/>
18. "Engineering genius." Anysphere. [Online]. Available: <https://anysphere.inc/>
19. "Magic." Magic.dev. [Online]. Available: <https://magic.dev/>
20. "Replit - Build apps and sites with AI." Replit. [Online]. Available: <https://replit.com/>
21. "Cody - Business AI employee trained on your knowledge base." Cody AI. [Online]. Available: <https://meetcody.ai/>
22. C. E. Jimenez et al., "SWE-bench: Can language models resolve real-world GitHub issues?" 2024, *arXiv:2310.06770*.
23. "CognitionAI/devin-swebench-results." GitHub. [Online]. Available: https://github.com/CognitionAI/devin-swebench-results/blob/main/output_diffs/fail/scikit-learn_scikit-learn-11542-diff.txt
24. "CognitionAI/devin-swebench-results." GitHub. [Online]. Available: https://github.com/CognitionAI/devin-swebench-results/blob/main/output_diffs/fail/scikit-learn_scikit-learn-25744-diff.txt
25. Z. Fan, X. Gao, M. Mirchev, A. Roychoudhury, and S. H. Tan, "Automated repair of programs from large language models," 2022, *arXiv:2205.10583*.
26. I. Bouzenia, P. Devanbu, and M. Pradel, "RepairAgent: An autonomous, LLM-based agent for program repair," 2024, *arXiv:2403.17134*.
27. C. Qian et al., "Iterative experience refinement of software-developing agents," 2024, *arXiv:2405.04219*.
28. Y. Shinyama, Y. Arahorri, and K. Gondow, "Analyzing code comments to boost program comprehension," in Proc. 25th Asia-Pacific Softw. Eng. Conf. (APSEC), 2018, pp. 325–334, doi: [10.1109/APSEC.2018.00047](https://doi.org/10.1109/APSEC.2018.00047).
29. W. Zhou, L. Zhang, L. Guan, P. Liu, and Y. Zhang, "What your firmware tells you is not how you should emulate it: A specification-guided approach for firmware emulation," in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., New York, NY, USA, 2022, pp. 3269–3283.
30. C. Oliveira, A. Aleti, L. Grunske, and K. Smith-Miles, "Mapping the effectiveness of automated test suite generation techniques," *IEEE Trans.*
- Rel., vol. 67, no. 3, pp. 771–785, Sep. 2018, doi: [10.1109/TR.2018.2832072](https://doi.org/10.1109/TR.2018.2832072).
31. "Introducing OpenAI o1-preview." OpenAI. [Online]. Available: <https://openai.com/index/introducing-openai-o1-preview/>
32. H. Muccini, P. Inverardi, and A. Bertolino, "Using software architecture for code testing," *IEEE Trans. Softw. Eng.*, vol. 30, no. 3, pp. 160–171, Mar. 2004, doi: [10.1109/TSE.2004.1271170](https://doi.org/10.1109/TSE.2004.1271170).
33. L. Zhang, P. Liu, Y.-H. Choi, and P. Chen, "Semantics-preserving reinforcement learning attack against graph neural networks for malware detection," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 2, pp. 1390–1402, Mar./Apr. 2022, doi: [10.1109/TDSC.2022.3153844](https://doi.org/10.1109/TDSC.2022.3153844).
34. N. T. Islam, J. Khouri, A. Seong, G. D. L. T. Parra, E. Bou-Harb, and P. Najafirad, "LLM-powered code vulnerability repair with reinforcement learning and semantic reward," 2024, *arXiv:2401.03374*.
35. R. Gupta, A. Kanade, and S. Shevade, "Deep reinforcement learning for syntactic error repair in student programs," in Proc. AAAI Conf. Artif. Intell., 2019, pp. 930–937, doi: [10.1609/aaai.v33i01.3301930](https://doi.org/10.1609/aaai.v33i01.3301930).
36. Q. Zou, L. Zhang, A. Singhal, X. Sun, and P. Liu, "Using explainable AI for neural network-based network attack detection," *Computer*, vol. 57, no. 5, 2024, pp. 78–85, doi: [10.1109/MC.2023.3342602](https://doi.org/10.1109/MC.2023.3342602).
37. Z. Niu, G. Zhong, and H. Yu, "A review on the attention mechanism of deep learning," *Neurocomputing*, vol. 452, pp. 48–62, Sep. 2021, doi: [10.1016/j.neucom.2021.03.091](https://doi.org/10.1016/j.neucom.2021.03.091).

LAN ZHANG is an assistant professor at the School of Informatics, Computing, and Cyber Systems, Northern Arizona University, Flagstaff, AZ 86005 USA. Contact her at lan.zhang@nau.edu.

ANOOP SINGHAL is a senior computer scientist in the Computer Security Division at the National Institute of Standards and Technology,

Gaithersburg, MD 20899 USA. Contact him at anoop.singhal@nist.gov.

QINGTIAN ZOU is a postdoctoral researcher at the University of Texas, Southwestern Medical Center, Dallas, TX 75390 USA. Contact him at qingtian.mill.zou@gmail.com.

XIAOYAN SUN is an associate professor with Department of Computer

Science, Worcester Polytechnic Institute, Worcester, MA 01609 USA. Contact her at xsun7@wpi.edu.

PENG LIU is the Raymond G. Tronzo, MD Professor of Cybersecurity and serves as the director of the Cyber Security Lab at The Pennsylvania State University, State College, PA 16803 USA. Contact her at pxl20@psu.edu.

Get Published in the *IEEE Open Journal of the Computer Society*

Get more citations by publishing with the *IEEE Open Journal of the Computer Society*

Your research on computing and informational technology will benefit from 5 million unique monthly users of the *IEEE Xplore®* Digital Library. Plus, this journal is fully open and compliant with funder mandates, including Plan S.



Submit your paper today!
Visit www.computer.org/oj to learn more.





Embracing Neurodiversity in Cybersecurity: What Lies Ahead?

Bilal Naqvi¹, Lappeenranta-Lahti University of Technology

Joakim Kävrestad¹, Jönköping School of Engineering

This article highlights the importance of neurodiversity in cybersecurity, explores the challenges in research and practice, and presents actionable outcomes for improvement in the state of the art.

With the digitalization across multiple sectors, including e-government, banking and finance, and health care, the need for effective security measures to protect the offered services is paramount. However, it is also important to consider how the users of these services interact with

cybersecurity functions because even technically robust solutions are susceptible to human errors leading to security breaches where the users' intentions cannot be blamed. It is worth mentioning that there has been progress in terms of consideration of human aspects of cybersecurity, and it has enabled consideration of the usability of the security mechanisms; however, the inclusivity concerns remain largely unaddressed.¹ This has implications for the users who cannot use the systems and services as normally abled users can. For instance,

it is unfair to blame a challenged user, who tried (to the best of their ability) to abide by the security recommendations, but their action still resulted in a breach. Therefore, knowing who the users are and considering inclusivity concerns is pertinent.¹

Inclusivity is a diverse concern with all its facets having the potential to impact the security of the systems and services. The IEEE 25010:2023 standard on Systems and software engineering—Systems and Software Quality

Digital Object Identifier 10.1109/MC.2025.3555463
Date of current version: 27 June 2025



Requirements and Evaluation (SQuaRE) defines inclusivity as the “capability of a product to be utilized by people of various backgrounds” and as additional note concerning the backgrounds, the standard refers to people of different age groups, abilities, cultures, ethnicities, languages, economic situations, literacy, and life situations.² These backgrounds are quite diverse, and each of these refers to multiple conditions, for instance, abilities refer to both cognitive and physical abilities. This article advocates for the noble goal of “inclusivity in cybersecurity” by considering all of the aspects and conditions mentioned earlier; however, the focus of the current article is on the impact of cognitive abilities on users’ security interactions, representing one facet of inclusivity. This is motivated by the fact that as per the reported stats, the number of people experiencing cognitive disabilities is higher than any other form of disability.³

In cognitive science, neurodiversity is a term often used to refer to unique ways a human brain works. Being neurodivergent means having brain functions different from the average “neurotypical” people.⁴ Thus, the term *neurodivergent* refers to people with cognitive challenges and limitations; these challenges are either by birth or developed during their lifetime.⁴ Within neurodiversity, there are several unique conditions (also referred to as cognitive disabilities) such as intellectual disabilities, autism spectrum disorders (ASD), attention-deficit/hyperactivity disorder (ADHD), mental illness, brain injury, and stroke, among others.³ Cognitively demanding cybersecurity tasks are “usability hinders” for the users, and these hindrances are often more severe for users with cognitive disabilities.⁵ A task that is perceived as a nuisance by a neurotypical user may be impossible for a user with a cognitive disability to complete.

COGNITIVE INCLUSIVITY AND CYBERSECURITY

Cognitive inclusivity is a cybersecurity concern as long as cybersecurity is dependent on users’ ability to correctly use cybersecurity functions. The cybersecurity functions should not only be usable by neurotypical users but also by the large population of neurodivergent users. There is a consequent need to understand how to address cognitive inclusivity in the development of cybersecurity functions. An intuitive way to do that is to look at the typical effects of different cognitive conditions. The Encyclopedia of the Neurological Sciences⁶ suggests

which stresses that the manifestation of cognitive conditions, especially in cases with comorbidity, is very individual.⁵ We, therefore, argue that targeting individual conditions is not the most effective way; and that cybersecurity functions must be developed with cognitive inclusivity with regard to cognitive functions rather than the conditions that impact these functions. This would entail discussing cognitive inclusivity in terms of cognitive functions, memory, learning, planning, reasoning, decision-making, and problem-solving. As an example in this regard, two comorbid users, both with ADHD and ASD might need different solutions to support

It is unfair to blame a challenged user, who tried (to the best of their ability) to abide by the security recommendations, but their action still resulted in a breach.

that, for instance, people with ADHD typically struggle with short attention spans and impulsivity, which is known to be related to cybersecurity behaviors such as an inability to concentrate on cybersecurity awareness training. In contrast, dyslexia is often manifested as writing difficulties which in cybersecurity can be attributed to the difficulties in managing passwords as a possible consequence. Furthermore, autism is associated with communication difficulties, which in cybersecurity terms can impede the users’ ability to understand security instructions and policies.

At closer inspection, the manifestation of cognitive disabilities appears to be more complex. The ways that different conditions are manifested differ from person to person. Furthermore, *comorbidity*, meaning that one individual is impacted by more than one condition, is common.⁷ This has been explored in cybersecurity research

their cybersecurity interactions relevant to affected cognitive functions, which in the case of the first user may be memory and learning, and for the second user could be decision-making and reasoning. Therefore, considering the uniqueness of conditions we must consider the cognitive functions in the design and development of cybersecurity functions. To illustrate this further, we exemplify (in Table 1) how memory, learning, and problem-solving can be associated with common security functions such as authentication, security training and awareness, and phishing detection.

As exemplified in Table 1, cognitive functions are central in common cybersecurity interactions. Unless the cognitive functions are properly addressed in the design and development of the platforms where the interactions take place, we argue that secure behavior for all users remains unattainable.

TABLE 1. Cognitive functions and their role in cybersecurity.

		Cognitive functions		
		Memory	Learning	Decision-making
Security functions	Authentication	Memorization of passwords	Learning multifactor authentication	Deciding what authentication type to use
	Security training and awareness	Memorization of new information	Learning new concept	Deciding what training modules to follow
	Phishing detection	Rely on remembering contacts to know if an email is from an unknown sender	Learning phishing indicators	Deciding if an email is legitimate or not

CHALLENGES IN THE STATE OF THE ART

Despite the importance of neurodiversity in cybersecurity, state-of-the-art research has identified several challenges that remain to be properly addressed. First, there needs to be a focus on divergent needs when designing and developing cybersecurity functions. This includes, for instance, older adults having memory issues, who have to create and remember the same complex passwords as an average user.⁸ Furthermore, when working with a neurodiverse population,

allowing wheelchair access. Courtesy of such directives, contemporary public spaces have accessible pathways. Therefore, there is a need for policy directives by different governmental and regional organizations that bind the development organization to consider cognitive inclusivity in the cybersecurity functions they develop, to thereby enforce a digital inclusive pathway.

Third, industrial practices have developed oblivious to the fact that the products they develop will be used by users with cognitive disabilities.

earlier, there needs to be research on the verification and validation of cybersecurity functions developed for neurodivergent users. In case of severe limitations, focus groups consisting of medical specialists, psychologists, and special education teachers, for instance, need to be conducted to gather requirements for divergent users, and that still poses a research challenge.

FUTURE DIRECTIONS

Cognitive inclusivity is a fundamental part of a democratic, inclusive, and secure cyberspace. This is not only a research dimension but also an inclusivity concern as identified in the sustainable development goals. As has been discussed earlier in this article, the importance of the topic is hardly controversial but requires research initiatives, and policy directives that support inclusivity in cybersecurity. To ensure equal access to technology for neurodiverse user groups, we suggest the following actions to be taken in research, practice as well as in policy development.

Research initiatives

Renuaud and Coles-Kemp¹ discuss the step toward inclusive and accessible cyberspace in detail. The authors suggest starting with defining the basics including who the users are, and what their abilities are, and then designing the cybersecurity functions with these concerns in mind. The authors also stress providing alternative options for performing a cybersecurity task such as providing both visual and audible

This is particularly important in the case of cognitive conditions impeding users' cognitive functions such as learning and decision-making.

an emphasis needs to be laid on considering the additional ethical challenges such as sufficiently conveying information about the study and gathering consent.⁸ This is particularly important in the case of cognitive conditions impeding users' cognitive functions such as learning and decision-making. Additional challenges in this regard can emerge in cases where assistive technologies are required for communication purposes.

Second, there is a lack of policy directives that make it compelling for development organizations to consider the cognitive aspects of cybersecurity.⁹ In several other sectors, policies have dictated consideration of accessibility concerns, for instance, public spaces must have accessible pathways

Consequently, cognitive disabilities are not considered during the design and development of cybersecurity functions.³ Within the industry, despite the realization of its importance, there is a lack of awareness on the topic, and the security practitioners are unaware of what needs users with cognitive disabilities have, as such needs are seldom specified by the clients.³

Finally, from an academic research perspective, there is a scarcity of research studies that consider cybersecurity and neurodiversity.¹⁰ Several open research questions remain: How can software requirements concerning neurodiversity in cybersecurity be gathered by involving users? In addition to the ethical perspective discussed

options to pass CAPTCHAs. Furthermore, there is a need to develop accessibility guidelines for the developers to support them in developing cybersecurity functions in similar contexts; in addition, this must be supported by developing heuristics for expert evaluation. Finally, dissemination venues for inclusive cybersecurity can help disseminate the research findings to a broader audience.

Furthermore, to enable equal opportunities for neurodivergent users, more research studies on identifying the needs of the users are required. We need to understand that one-fits-all cybersecurity solutions do not serve the neurodivergent users' needs. For conducting such studies, the proposal of "ethics in practice"⁸ is an important consideration to ensure informed consent and effective communication of the study's aims and objectives to the participants.

Practical initiatives (for industry)

As mentioned earlier, the state of the art in the industry reveals a lack of understanding of the needs of neurodivergent users. Interdisciplinary collaboration with domains such as medicine, social science, and humanities can help understand divergent user needs. Besides understanding the needs and gathering requirements, such forums can also prove to be beneficial for raising awareness on the topic. Such collaboration initiatives can also train software developers and security engineers on neurodiversity concerns.

In addition to consensus on the importance of neurodiversity in cybersecurity, the practitioners must take concrete steps to implement the same in the products they develop. The developmental practices concerning cybersecurity functions must evolve to consider inclusivity concerns in addition to robustness of security and usability.

Policies initiatives

There are several instances from the past where policy directives improved

the state of the art for instance in the case of accessible public spaces discussed earlier. Therefore, like other domains where inclusivity and accessibility are established principles, there need to be policy directives that bind the development organizations to consider neurodivergent needs in the design and development of cybersecurity functions. It is relevant to mention that ISO 25010:2023² is a step in this direction, which lists inclusivity as one of the characteristics of the product quality model.

To conclude, the article discussed neurodiversity in relation to cybersecurity, which is related to the overall goal of inclusive cybersecurity. As said, inclusivity is a diverse concern, and it requires initiatives across the policymakers, researchers, and practitioners to address cognitive inclusivity in the development of future cybersecurity functions. Ensuring that, to the greatest extent possible, cybersecurity functions can be used by all people, regardless of their cognitive ability, is an obvious precursor for equal access to technology. □

REFERENCES

1. K. Renaud and L. Coles-Kemp, "Accessible and inclusive cyber security: A nuanced and complex challenge," *SN Comput. Sci.*, vol. 3, no. 5, Jun. 2022, Art. no. 346, doi: [10.1007/s42979-022-01239-1](https://doi.org/10.1007/s42979-022-01239-1).
2. Systems and Software Engineering—Systems and Software Quality Requirements and Evaluation (SQuaRE)—Product Quality Model, ISO/IEC and I. 25010:2023, International Organization for Standardization, Geneva, Switzerland, 2023.
3. B. Naqvi, J. Kävrestad, and A. K. M. Najmul Islam, "Inclusive and accessible cybersecurity: Challenges and future directions," *Computer*, vol. 57, no. 6, pp. 73–81, Jun. 2024, doi: [10.1109/MC.2024.3376827](https://doi.org/10.1109/MC.2024.3376827).
4. N. Baumer and J. Frueh, "What is neurodiversity?" *Harvard Health Blog*, Nov. 23, 2021. Accessed: Mar. 11, 2025. [Online]. Available: <https://www.health.harvard.edu/blog/what-is-neurodiversity-202111232645>
5. J. Kävrestad, J. Rambusch, and M. Nohlberg, "Design principles for cognitively accessible cybersecurity training," *Comput. Secur.*, vol. 137, Feb. 2024, Art. no. 103630, doi: [10.1016/j.cose.2023.103630](https://doi.org/10.1016/j.cose.2023.103630).
6. R. B. Daroff and M. J. Aminoff, *Encyclopedia of the Neurological Sciences*. Oxford, U.K.: Academic Press, 2014.
7. D. Dewey, "What is comorbidity and why does it matter in neurodevelopmental disorders?" *Curr. Develop. Disorders Rep.*, vol. 5, no. 4, pp. 235–242, 2018, doi: [10.1007/s40474-018-0152-3](https://doi.org/10.1007/s40474-018-0152-3).
8. F. Sharevski, "Inclusive involvement of at-risk users in cybersecurity research," *IEEE Security Privacy*, vol. 22, no. 5, pp. 13–22, Sep./Oct. 2024, doi: [10.1109/MSEC.2024.3416878](https://doi.org/10.1109/MSEC.2024.3416878).
9. P. D. Chowdhury and K. Reanud, "Advocating a policy push toward inclusive and secure "digital-first" societies," *IEEE Security Privacy*, vol. 22, no. 5, pp. 23–31, Sep./Oct. 2024, doi: [10.1109/MSEC.2024.3431278](https://doi.org/10.1109/MSEC.2024.3431278).
10. J. Kävrestad and B. Naqvi, "Cognitively available cybersecurity: A systematic literature review," in *Proc. Int. Conf. Human-Centred Softw. Eng. (HCSE)*, Cham, Switzerland: Springer-Verlag, 2024, doi: [10.1007/978-3-031-64576-1_9](https://doi.org/10.1007/978-3-031-64576-1_9).

BILAL NAQVI is a postdoctoral researcher at Lappeenranta-Lahti University of Technology, 53850 Lappeenranta, Finland. Contact him at syed.naqvi@lut.fi.

JOAKIM KÄVRESTAD is an Assistant Professor in Computer Science at Jönköping School of Engineering, 553 18 Jönköping, Sweden. Contact him at joakim.kavrestad@ju.se.



Hype Me to the Moon! Let Me Dream Among the AI Stars! Let's Give up on That Guy Who Wants to Go to Mars ...

Michael Zyda^{ID}, University of Southern California

Is the generative artificial intelligence business in a hype cycle, or are we really going somewhere with it all other than a narcissistic trip to a distant planet?

In a previous column,¹ I spoke about the artificial intelligence (AI) hype cycle and how it “historically ended with unrealistic expectations being held and disappointments to funders (research sponsors, venture capitalists, and corporate investments), with all of this

resulting in funding cuts and researchers and investors abandoning the area.” Now, we don’t really know if we are in a hype cycle (Figure 1), but there are many signals that indicate that we may be and, if we are, we need to know so that we can plan for a smoother landing. And, to top it all off, we are in the midst of a very large self-inflicted financial crisis that was completely avoidable.

Right now, megacompanies, the so-called magnificent seven (Apple, Alphabet/Google, Amazon, Meta Platforms, Nvidia, Microsoft, and Tesla), are spending big bucks

on data centers running giga-numbers of power-sucking Nvidia machines as if that was the path to the truth, the way and the light. If we are in a hype cycle, then we need to call it out, especially since our AI stars are building these centers in locations short on power and where there is a delicate and fragile power infrastructure (Texas, etc.) that currently can barely provide power for heating and cooling for the people that live there now! Many of these

Digital Object Identifier 10.1109/MC.2025.3561538
Date of current version: 27 June 2025

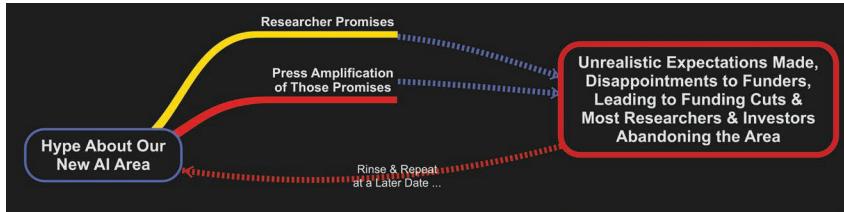


FIGURE 1. The AI hype cycle.

places are also far away from universities capable of graduating engineers that can actually build and run said data centers.

Is the economic crisis going to limit our ability to fly us to the moon to dream among the AI stars? (Figure 2)

Well, Nvidia stock was going to the moon, and now it's below the 2024 stock-split level, despite the promises of megacompanies to purchase millions of graphics processing units (GPUs) at a time!

We see Microsoft planning on investing US\$80 billion into AI data centers in 2025. Additionally, we see Microsoft signing a 20-year deal with Constellation Energy to purchase power from the

renovated Three-Mile Island nuclear power plant, with Constellation spending US\$1.6 billion to restart the facility.

We see Google investing US\$75 billion into AI data centers in 2025 after having spent US\$52.5 billion in 2024 on AI and cloud infrastructure.

We see Meta Platforms investing US\$65 billion in AI infrastructure in 2025, including a massive AI data center in Louisiana. Meta is also discussing a US\$200 billion AI data center in states like Louisiana, Wyoming, and Texas.

Amazon (AWS) is investing over US\$100 billion in 2025 to boost its AI and cloud infrastructures, including building new data centers and

COMMENTS?

If you have comments about this article, or topics or references I should have cited or you want to rant back to me on why what I say is nonsense, I want to hear. Every time we finish one of these columns, and it goes to print, what I'm going to do is get it up online and maybe point to it at my Facebook (mikezyda) and my LinkedIn (mikezyda) pages so that I can receive comments from you. Maybe we'll react to some of those comments in future columns or online to enlighten you in real time! This is the "Games" column. You have a wonderful day.

upgrading existing ones to support AI services.

We see xAI is part of a US\$30 billion fund with Microsoft and Blackrock, an

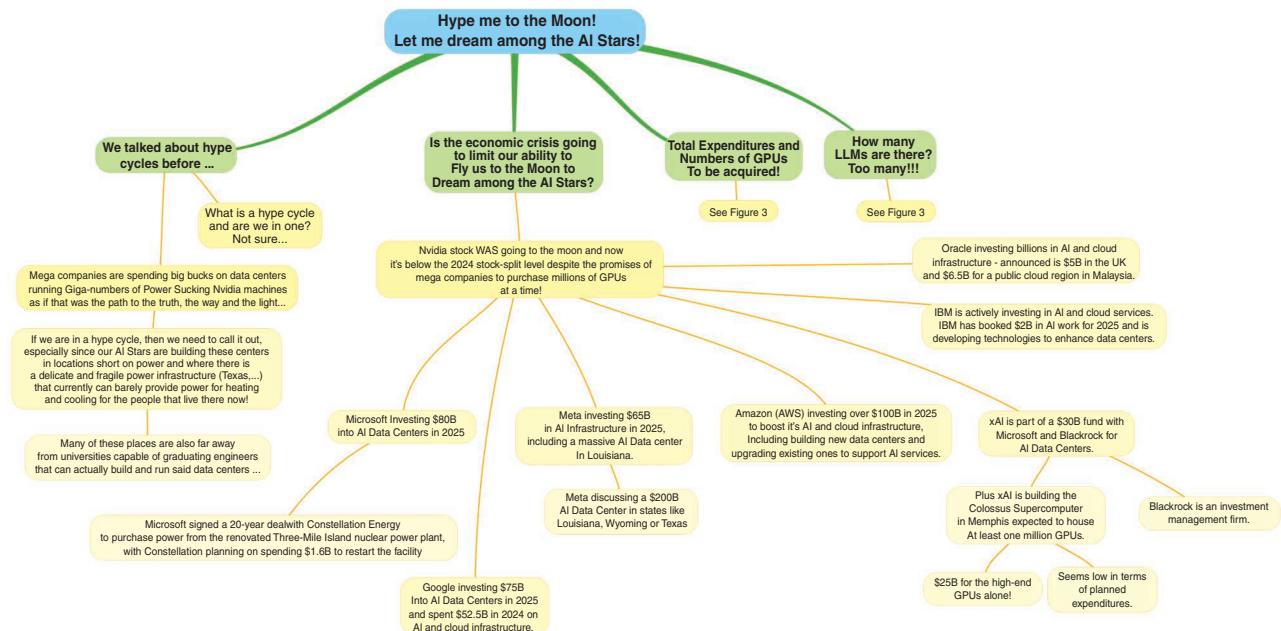


FIGURE 2. Hype me to the moon! Let me dream among the AI stars!

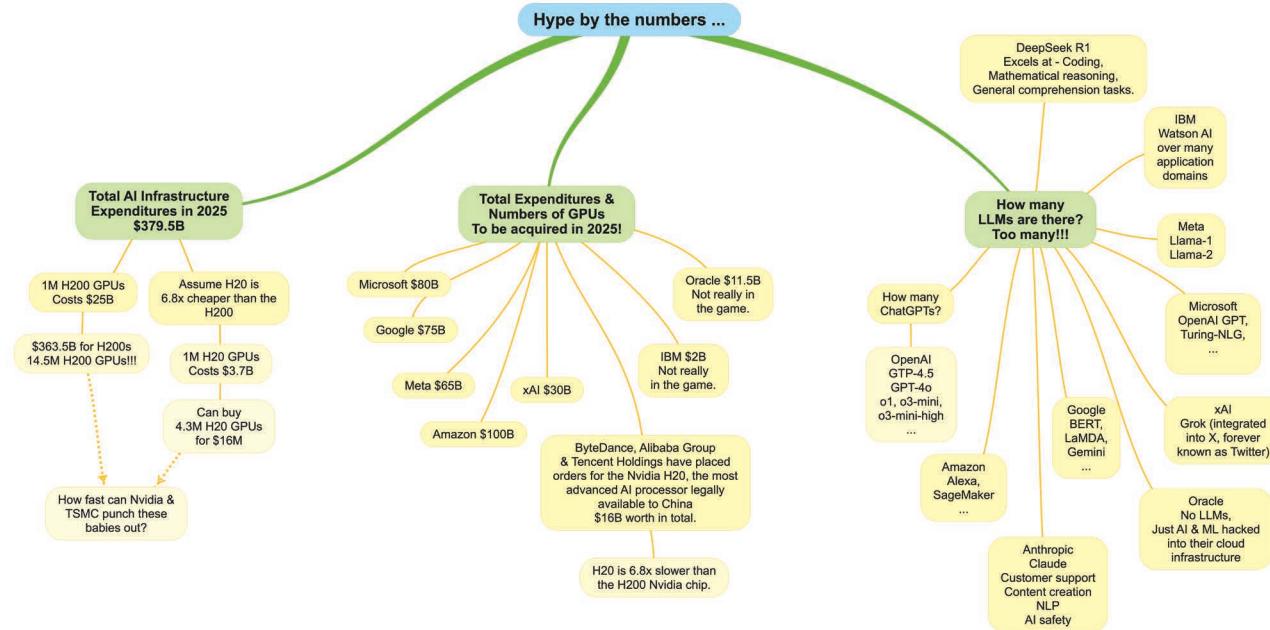


FIGURE 3. Hype by the numbers.

investment management firm, for AI data centers. Plus, xAI is building the Colossus Supercomputer in Memphis, which is expected to house at least one million GPUs, with US\$25 billion being spent on the high-end GPUs alone! xAI's planned expenditures seem low in comparison to some of the other megacompanies in this space.

Other companies dabbling in this space are Oracle, which is investing

but not see large expenditures. So, let's look at total expenditures and the numbers of GPUs to be acquired in 2025. In Figure 3, we see Microsoft (US\$80 billion), Alphabet/Google (US\$75 billion), Meta Platforms (US\$65 billion), Amazon (US\$100 billion), xAI (US\$30 billion), ByteDance/Alibaba group/Tencent Holdings (US\$16 billion), IBM (US\$2 billion), and Oracle (US\$11.5 billion). So, total AI infrastructure expenditures in 2025 is

Holdings can purchase some 4.3 million H20 GPUs for their US\$16 million investment.

So, the real question then becomes how fast can Nvidia and Taiwan Semiconductor Manufacturing Corporation punch these GPUs out? Can they even make them all in 2025? We asked ChatGPT^a what was Nvidia's production capacity in 2025, and it said no number was released for 2024 or 2025. ChatGPT did state that in 2023, Nvidia shipped around 3.76 million data center GPUs, which is way short of the 14.5 million H200s being ordered in 2025. The 3.76 million number is about the number that the three Chinese AI companies require, 4.3 million H20 GPUs.

So, in order for Nvidia to meet the planned 14.5 million GPU orders for 2025, it has to have increased its production capacity by 3.9 times over its 2023 production capability! Wow! Lots of demand but four times too small a production capacity in 2023, but maybe reasonable in 2025! So, we are not hype, but with the market crash and the current economic mess, we

If we are in a hype cycle, then we need to call it out.

in AI and cloud infrastructure. US\$5 billion of that investment is in the United Kingdom and US\$6.5 billion is in Malaysia for a public cloud region.

IBM is actively investing in AI and cloud services. IBM has booked US\$2 billion in AI work for 2025 and is developing technologies to enhance data centers.

HYPE BY THE NUMBERS

Well, if we have hype rather than big business development, we would expect to hear a lot of noise and wild promises

about US\$379.5 billion, not a small number; in fact, it is a very nice number in terms of infrastructure spending.

We want to figure out how many GPUs this is going to buy. We already know that 1 million Nvidia H200 GPUs is about US\$25 billion. If we subtract out the Chinese GPU investment, this is US\$363.5 billion for H200s or 14.5 million H200 GPUs!

China is restricted to the purchase of H20 GPUs, which are 6.8 times slower than the H200 Nvidia GPUs. So, ByteDance/Alibaba Group/Tencent

^aAll of the numbers in AI infrastructure investments were searched for using ChatGPT 4.0 on 4 April 2025.

don't see this easily happening as the placement of money by companies onto the GPU production bus may no longer be possible.

HOW MANY LARGE LANGUAGE MODELS ARE THERE?

There are way too many large language models (LLMs)! I am not sure that it is possible for me to count them all. I listed the obvious ones, the ones from the megacompanies in the news (Figure 3).

OpenAI is the starting point, and the question to ask is, how many ChatGPTs are there? It seems there is a new one with a special purpose almost every week, and most people I know are using ChatGPT of some version. The biggest issue with ChatGPT though is its legal problem over the licensing of the training data it used in creating its GPT line of models.

Amazon has Alexa and SageMaker for their "AI models," and they are very primitive—they are nowhere near ChatGPT's capabilities. Amazon does offer Anthropic's Claude models on AWS. OpenAI's GPT models are not available on AWS. AWS does offer its Amazon Nova and Amazon Titan AI models on AWS as well as other foundation models through Amazon Bedrock.

Anthropic is the one most people ask me about with respect to investing. Anthropic's Claude LLM is listed as good for customer support, content creation, natural language processing, and AI safety. So, it seems it has several great reasons for its existence and usage.

Google has its own model, Gemini, and there is also Gemini Pro. Gemini is good for natural language understanding, conversation, and other capabilities. In my experience, Gemini seems less capable than ChatGPT, and I have found that often Gemini gives me made-up or just plain wrong answers for the question I asked. Besides Gemini, Google has other LLMs, BERT, LaMDA, and others built internally as part of their research efforts.

xAI has Grok, which is integrated into X (forever known as Twitter). xAI's Grok most likely will die as the CEO has stopped paying attention to his quiver of companies—hopefully he will be successful in flying to Mars. The Grok model focuses on real-time information retrieval and conversation. Grok is "designed to be helpful, honest, and harmless, integrating data from X (forever known as Twitter)." We all laughed at that statement, so I had to put it into this article.

Microsoft licenses OpenAI's GPT series of models and additionally has its own Turing-NLG. Turing-NLG is a powerful natural language generation model known for its ability to generate human-like text, understand context and assist in various applications, like chatbots, summarization, and content creation. Turing-NLG is a very capable model and will succeed as Microsoft is a megacompany with a long history of success.

Meta Platforms has its Llama series of models, which are known to be outstanding for natural language processing. The Llama series of models are available on Amazon AWS through Amazon Bedrock.

DeepSeek's R1 model excels at coding, mathematical reasoning, and general comprehension tasks. It is a Chinese model that has had some recent success. DeepSeek's R1 models are available on Amazon Bedrock. U.S. companies may avoid using it because it is from China, which is just the way it goes. That is sad.

IBM Watson AI works well and has been deployed by IBM over many application domains. IBM is mostly not in the news with respect to LLMs, but they have a long history of deploying technology quietly.

Oracle says it uses its LLMs embedded into their cloud infrastructure for enterprise solutions, but there is rare press about their LLM efforts.

So, lots of models to choose from which makes it difficult, but choosing OpenAI's GPT series, Google's Gemini once it has been cleaned up,

Anthropic's Claude, Microsoft's Turing-NLG, or Meta's Llama seem like reasonable choices. It comes down to what is your favorite company. That favorite company might just be Amazon AWS as it provides standardized access to almost all of the foundational models! Eventually, I believe these models will merge in capability and will all seem like the same thing, especially since we will most likely end up getting them through Amazon AWS, just like our groceries ...

Well, after all that, we see a great business in building GPU hardware to be placed in all of the promised data centers, but we don't know when all of those centers will receive their GPUs to be installed there. We see there are plenty of great LLMs, many with overlapping purposes and differing qualities of outputs that will be repaired over time. So, I want to say NOT on the hype label, except for statements by that CEO that needs to get to Mars soonest!!! ☺

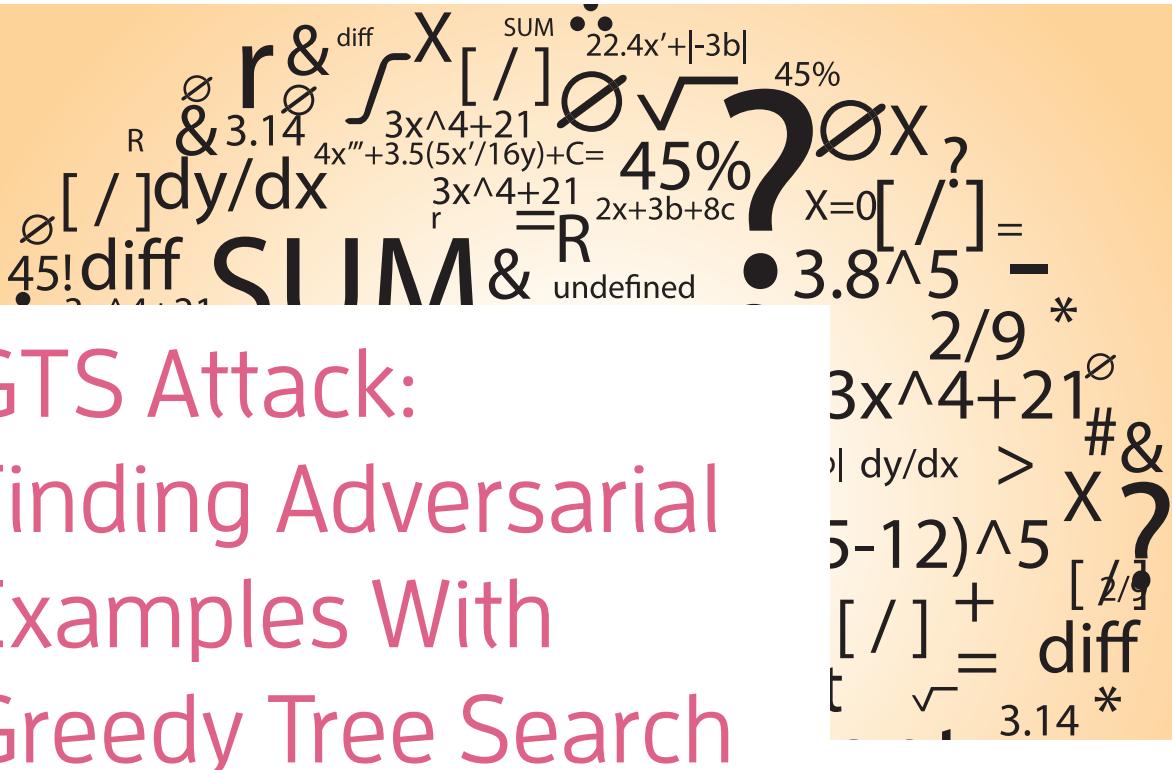
ACKNOWLEDGMENT

The author wishes to thank those readers who have gotten to the end of this bimonthly column without finding all of the deliberate and accidental errors. And I apologize for my personal commentaries, but I think they are very important for our futures. And I really do hope that a select section of mankind ends up on Mars!!!

REFERENCE

1. M. Zyda "Large language models and generative AI, oh my!," *Computer*, vol. 57, no. 3, pp 127–132, Mar. 2024, doi: 10.1109/MC.2024.3350290.

MICHAEL ZYDA is an emeritus professor of practice at the University of Southern California, Los Angeles, CA 90007 USA. Contact him at zyda@mikezyda.com.



GTS Attack: Finding Adversarial Examples With Greedy Tree Search

Christopher Clark and Armon Barton 

This article presents a novel approach for exposing the vulnerability of deep neural networks to adversarial examples, particularly in the context of a black-box machine learning-as-a-service environment, and we introduce the greedy tree search attack.

Deep learning has led to breakthroughs in a wide variety of domains including health care,¹² natural language processing,² and computer vision.³ Unfortunately, deep neural networks (DNNs) are vulnerable to adversarial examples,¹⁷ imperceptibly small pixel perturbations that can deceive DNNs.^{5,10,9,22} In the research, many threat models have

emerged including the white-box assumption where the adversary has knowledge and access to a model parameters and gradient.¹⁴ However, in machine learning (ML) as a service (MLaaS), the adversary likely will not have access to the model parameters or gradient.²¹ In the black-box setting where an adversary does not know the model gradients, the three prominent threat models are decision-based, score-based, and transfer-based.⁴ For decision-based, the adversary queries the model to observe the discrete class prediction.⁴ In score-based, the adversary queries the model to observe the class probabilities.¹ In transfer-based, the adversary crafts adversarial examples on a surrogate model followed by transferring them to the target model.¹⁴

While decision-based and score-based attacks are more realistic in real-world applications, such as MLaaS, many attacks in the literature rely on excessive model queries that may be easily evaded by the defender.⁶ On the other hand, attacks that are designed under the white-box



assumption do not transfer very well or can be defended against by the target model under the transfer-based black-box assumption.¹⁸

In this article, we are the first to leverage a greedy tree search (GTS) algorithm to find adversarial perturbations—called GTS attack—in which cross-entropy loss is used as a heuristic to search for adversarial examples in the input space. With minor hyperparameter tuning, the GTS attack achieves competitive performance under the white-box setting and state-of-the-art performance under the decision-based black-box setting in which we show top attack success rates with minimal queries compared to other top query-based attacks in the literature. We evaluate GTS attack on CIFAR10 and compare the performance against other state-of-the-art attacks including Carlini & Wagner (CW),⁵ Projected Gradient Descent (PGD),¹¹ AutoAttack,⁸ HopSkipJump Attack (HSJA),⁶ and Efficient Combinatorial Optimization Attack (ECO),¹³ and defenses including adversarial training (AT),¹¹ gradient regularization (GR),¹⁵ and TRadeoff-inspired Adversarial DEfense via Surrogate-loss minimization (TRADES).²⁰

METHODOLOGY

We propose a novel attack called the GTS attack that includes three settings: a white-box attack, a black-box score-based attack, and a black-box hybrid attack. Each attack has an optimization for L_{\inf} , L_2 , and L_0 and can be targeted or untargeted. Each attack setting was tested on multiple datasets with remarkable results. Specifically, the settings are called GTS White-Box Attack, GTS Black-Box Score-Based Attack, and GTS Black-Box Hybrid Attack which has two variations depending on if the substitute model is white-box or black-box. The substitute model, denoted by model_s , is used to generate

adversarial examples for the GTS Black-Box Hybrid Attack. Table 1 lists the attack names along with their description. The variants will be explained in greater detail later in this section.

GTS algorithm

GTS is an informed search algorithm commonly used in artificial intelligence applications for finding near-optimal paths efficiently.^{16,19} Its cost function is denoted by $h(n)$ known as the heuristic, which is the estimated cost from node n to a goal state.

Figure 1 shows a graph where a path exists from node A to node G. Each node is connected by an edge and has a red number representing $h(n)$ which is an estimated cost to reach node G from

any node n . For example, the estimated cost $h(D)$ to reach G from D is equal to 5. The black numbers represent the actual path cost; hence, the real cost to reach G from D is equal to 7.

GTS uses a priority queue that places the node with the lowest $h(n)$ value at the top of the queue and explores the lowest cost nodes first. Figure 2 shows this process as nodes in the queue are sorted by their $h(n)$ values with the graph search being depicted as a tree. Node A is the start state and first node to be explored. The successor nodes, B and C, are added to the queue in priority order based on lowest $h(n)$ cost. Nodes in the priority queue that have not been explored are depicted as gray while explored nodes are black. Node C is explored next as seen in Step 1, followed by D in Step 2 where

TABLE 1. GTS attack names and descriptions.

Attack Name	Attack Type
GTS Attack _w	GTS White-Box Attack
GTS Attack _s	GTS Black-Box Score-Based Attack
GTS Hybrid _w	GTS Black-Box Hybrid Attack, white-box model _s
GTS Hybrid _s	GTS Black-Box Hybrid Attack, score-based model _s

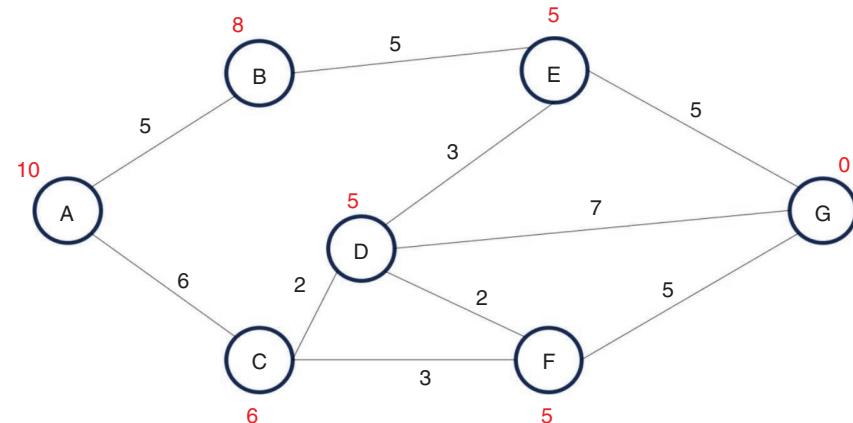


FIGURE 1. Example of a GTS graph. Nodes are represented as circles, edges as lines connecting nodes, and red numbers are $h(n)$ for that node. Black numbers are the true path cost between connected nodes.

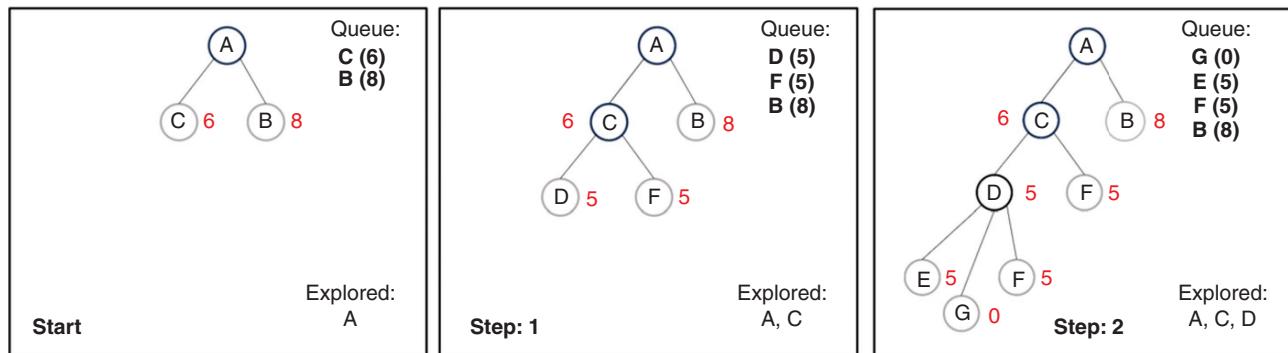


FIGURE 2. Example of GTS graph traversal. $h(n)$ is depicted in parenthesis next to the nodes in the queue.

the goal state is found and the algorithm terminates. Note that the goal state was found in two steps while exploring three nodes returning the path A-C-D-G with real cost of 15. An optimal search such as A* would return the path A-C-F-G with real cost of 14. However, finding the optimal path is more computationally expensive as A* would explore six nodes to find this path. GTS is slightly less optimal but significantly faster.

GTS attack algorithm

In the GTS graph, images are represented as nodes in a tree. The heuristic $h(n)$ is taken by applying cross-entropy loss to the current node n discussed in more

detail later. The node with the lowest $h(n)$ cost will be next out of the queue. A goal test determines if the image meets the adversarial example requirements. If it does, the algorithm ends returning the adversarial example. Otherwise, ψ successors are created by adding random perturbations to the current image followed by computing their respective $h(n)$ cost. Then, the successor nodes are added to the priority queue, and the image with the lowest $h(n)$ cost will be the next node out of the queue. As the graph is traversed, the deeper nodes will typically have more perturbation.

Since $h(n)$ is an estimated cost to reach the goal state, it should decrease

as the algorithm moves closer to finding an adversarial example. For untargeted attacks, the score of the true class label prediction is used for $h(n)$ since the score decreases as the adversarial example moves closer toward the decision boundary. We use the softmax probability, although it seems reasonable that logits would work if properly normalized. For targeted attacks, the negative score of the target class prediction is used for $h(n)$ since the score is being maximized toward the target class.

Figure 3 represents the structure of the attack. The first node is the original image of a “Cat.” Two successor nodes are generated with random perturbations (perturbations are significantly enhanced for visibility) where $h(n)$, shown in red, represents the model’s prediction probability for that image’s true label. The search algorithm prioritizes nodes in which the perturbation has the greatest effect on minimizing the true label score thereby finding examples that are shallow within the tree, and thus, with minimal perturbation.

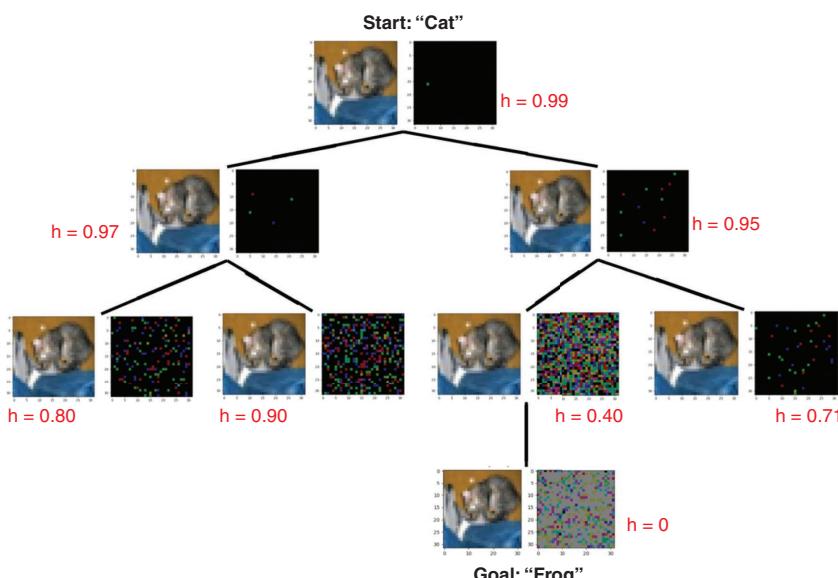


FIGURE 3. Visualization of the GTS attack process. The tree begins with an original input image as the start state and traverses the path created through pixel perturbations to an adversarial example as the goal state.

GTS White-Box Attack

The GTS White-Box Attack (GTS Attack_w) is the default attack as shown in Algorithm 1.

In lines 1–4, the custom-bounded queue q is initialized, $h(n)$ is set to zero, the first node tuple n is initialized with a benign input sample, and the explored set e is initialized to zero. Line 5 adds the first node to the queue. Line 6 begins the main while loop to generate the adversarial example as long as

nodes are in q . In line 7, the node with the lowest $h(n)$ value is popped from the queue. The node attributes are assigned to the corresponding variables in lines 8–9; X_{adv} is the current sample, and \hat{y} is the model prediction on the current sample. α is periodically increased in line 10 based on how long the attack has run. The potential adversarial example is hashed in line 11, and the node is only explored if it has not already been explored per line 12. The hash value is added to the explored set in line 13. Lines 14–16 are the goal test to see if the adversarial example is successful. If the goal is met, the algorithm terminates. For targeted attacks, line 14 will check that the prediction matches the target label. The number of pixels to perturb, r , is selected at random in line 17 from a uniform distribution. The sign of the gradient, denoted by τ , is calculated in line 18.

The **for** loop in line 19 expands the current node to create the successor nodes. Line 20 calls the function to create the perturbations which is described in detail later. The successor image, X_{child} , is created by adding the perturbation to X_{adv} in line 21.

Line 22 returns the softmax probabilities of the model with respect to the input sample X_{child} , and line 23 returns the predicted class \hat{y} . The heuristic $h(n)$ is computed in line 24. In an untargeted attack, $h(n)$ is the score of the true class causing the algorithm to drive perturbations in the direction that decreases the model confidence in the true class. When conducting a targeted attack, line 24 is modified so $h(n)$ is the negative score of the target class driving perturbations in the direction that increases the model confidence in the target class. Line 25 creates the successor node, and line 26 adds the successor to the priority queue. The process continues until the goal state is achieved or the queue is empty.

GTS score-based Black-Box Attack

The GTS Score-Based Black-Box Attack (GTS Attack_s) follows the

same steps as previously outlined with the exception of calculating the model gradient in line 18 of

Algorithm 1. Instead, perturbation is randomly sampled from the normal distribution.

ALGORITHM 1: GTS attack

Variables

X : Sample image
 y : Truth label
 α : initial standard deviation for pixel intensity drawn from $N(0, \alpha)$
 ρ : dimensions of X : $height \times width \times channel$
 ψ : max branching factor, that is, number of successor nodes

Functions

$perturbations(r, \alpha, \tau)$: generates pixel perturbations
 $model.predict(x)$: produces model prediction
 $increase_alpha(\alpha)$: increases alpha periodically
 $argmax(array)$: calculates the max index value
 $loss$: calculates the loss with respect to a label and input
 $sign$: calculates the sign of the gradient
 $Bounded\ PriorityQueue(bound)$: bounded length priority queue

```

1:  $q \leftarrow Bounded\ PriorityQueue(bound)$ 
2:  $h(n) \leftarrow 0$ 
3:  $n \leftarrow (X, y)$ 
4:  $e \leftarrow set()$ 
5:  $q.add(h(n), n)$ 
6: while  $q$  do
7:    $n \leftarrow q.get()$ 
8:    $X_{\text{adv}} \leftarrow n.X$ 
9:    $\hat{y} \leftarrow n.\hat{y}$ 
10:   $\alpha \leftarrow increase\_alpha(\alpha, len(e))$ 
11:   $X_{\text{hash}} \leftarrow X_{\text{adv}}.\text{hash}()$ 
12:  if  $X_{\text{hash}}$  not in  $e$  then
13:     $e.add(X_{\text{hash}})$ 
14:    if  $\hat{y} \neq y$  then
15:      return  $X_{\text{adv}}$ 
16:    end if
17:     $r \leftarrow randint(1, \rho)$ 
18:     $\tau \leftarrow sign(\nabla_{X_{\text{adv}}} loss(y, \hat{y}))$ 
19:    for  $i$  in  $range(\psi)$  do
20:       $\delta \leftarrow perturbations(r, \alpha, \tau)$ 
21:       $X_{\text{child}} \leftarrow X_{\text{adv}} + \delta$ 
22:       $y_{\text{pred}} \leftarrow model.predict(X_{\text{child}})$ 
23:       $\hat{y} \leftarrow argmax(y_{\text{pred}})$ 
24:       $h(n) \leftarrow y_{\text{pred}}[y]$ 
25:       $n \leftarrow (X_{\text{child}}, \hat{y})$ 
26:       $q.put(h(n), n)$ 
27:    end for
28:  end if
29: end while
```

GTS Hybrid Black-Box Attack

The GTS Hybrid Black-Box Attack is a novel black-box attack that produces state-of-the-art results with very few model queries on the target model. The attack requires a surrogate model denoted as model_s , and a target model denoted as model_t . To minimize queries on model_t , [Algorithm 1](#) queries model_s as needed while only querying model_t periodically. We define a hyperparameter μ to control the frequency of model_t queries such that for every μ queries to model_s , model_t gets queried once. To accomplish this, we set a counter, called queries_s to count model_s queries. Additionally, there is no need to query model_t until the adversarial example has reached the goal on model_s . Therefore, we introduce a Boolean variable, denoted as goal_s , that is initialized to False and is set to True when the adversarial example defeats model_s for the first time.

For attacking model_s in this procedure, the GTS Attack_w may be used to leverage the gradients of model_s , or GTS Attack_s may be used to solely use the prediction probabilities of model_s . In the experimental results, the former is denoted as GTS Hybrid_w, and the

latter by GTS Hybrid_s. It is important to note that in our implementation we set $h(n)$ to zero once goal_s is achieved, and every time model_t is queried to encourage the algorithm to keep traversing the same path that defeats model_s , and thereby quickly leads to misclassification by model_t . With this approach, we show in the evaluation that a CIFAR10 model can be attacked in 5 to 14 queries compared to 57 and 68 queries for HSJA and ECO attack respectively. On the other hand, while this hybrid approach is made possible by the GTS procedure, periodically querying the target model while attacking the surrogate is not straightforward for other attacks. For example, with PGD, once the surrogate model is defeated, there is no added benefit with continuing to traverse the gradient on the surrogate model. In the evaluation, we attack the surrogate model using PGD with up to 1,000 iterations before transferring to the target model, yet the transferability was 32% compared to 98% for GTS Hybrid Attack. Though the PGD transfer attack can be done in one query, the attack success rate (ASR) is much lower compared to the GTS Hybrid Attack.

Similar transferability results were found with other popular attacks such as C&W and AutoAttack.

Targeted attack

In a targeted white-box attack, we change line 18 of Algorithm A* to minimize the loss between the truth label and the target label. To determine the target class, we employ the class with the second highest probability, following the true class. Line 24 in [Algorithm 1](#) is modified to drive the perturbations toward the target class vice away from the truth label. This approach was consistently applied across all targeted attacks.

 L_p norm optimization

Up to this point, we have not said anything about optimizing for an L_p norm. It was observed that the GTS procedure finds adversarial examples with low perturbation relative to both L_{\inf} and L_2 norms without being explicitly forced to do so. This is due to the way in which perturbation is added during each iteration as described previously, and because the heuristic allows the algorithm to find

TABLE 2. CIFAR10: Untargeted attack results.

Vanilla				AT		GR		TRADES	
mean	ASR	mean	ASR	mean	ASR	mean	ASR	mean	ASR
L_2	A* Attack _w	0.17	100%	0.25	100%	0.26	100%	0.31	100%
	A* Attack _s	0.18	100%	0.25	100%	0.23	100%	0.36	100%
	CW	0.18	96%	0.25	100%	0.26	93%	0.33	90%
	PGD	0.20	80%	0.25	92%	0.25	70%	0.40	81%
	AutoAttack	0.30	89%	0.30	79%	0.30	89%	0.30	74%
L_{∞}	A* Attack _w	0.0044	100%	0.0062	100%	0.0071	100%	0.0090	100%
	A* Attack _s	0.0150	100%	0.0236	100%	0.0221	100%	0.0336	100%
	CW	0.0085	95%	0.0112	91%	0.0118	92%	0.0125	88%
	PGD	0.0100	55%	0.0400	35%	0.0399	45%	0.0195	38%
	AutoAttack	0.0080	89%	0.0100	87%	0.0100	86%	0.0100	75%
L_0	A* Attack _w	3.72	100%	5.30	100%	5.53	100%	4.18	100%
	A* Attack _s	3.73	100%	5.37	100%	5.58	100%	5.58	100%

examples that are relatively shallow in the search tree. Additionally, with minor changes, the GTS procedure can also find examples with low perturbation according to the L_0 norm. To achieve this, line 17 in [Algorithm 1](#) was modified to enforce that only one pixel may be changed during each iteration. Results for all three L_p norms are shown in the evaluation.

Time and space complexity

GTS is not optimal and has a worst-case time complexity of $O(|V|)$ where $|V|$ denotes the number of vertices or nodes within the search space. However, with a good heuristic function, the time complexity can be reduced to $O(bm)$, where b is the branching factor, or the average number of successors per node, and m is the maximum number of actions per path.¹⁶ The time complexity can be significant if the branching factor is large and the depth of the solution is deep. Without any mitigation, the adversarial example search space would be intractable. However, we apply pruning techniques by introducing a custom bounded queue, and by limiting the number of successor nodes with the tunable parameter ψ . For L_2 optimized attacks, ψ is set to a value between 3 and 10. In an L_0 setting, we used ψ values as high as 5,000 for CIFAR10. This form of pruning reduces time and space complexity allowing for reasonable run times. However, there are many goal states within the vast adversarial example search space; though most of the search space is pruned, GTS attack can still find near-optimal solutions as shown in the evaluation. Though GTS attack is not optimal, the tradeoff compared to A^* is that GTS finds near-optimal solutions while expanding significantly fewer nodes. However, near-optimal solutions are still successful adversarial examples. There are combinatorially many ways to add perturbation to a sample to fool the model, and GTS only needs to find one while A^* would be required to search a vast space to find an optimal solution.

GTS still relies on an admissible heuristic to find near-optimal solutions. In our application, the inverse cross-entropy or inverse of the highest logit is a small number—significantly smaller than the number of steps to reach the goal.

RESULTS

Untargeted attacks

[Table 2](#) shows the untargeted GTS Attack results on the CIFAR10 dataset against four different models with varying levels of defense. Our L_{inf} reported results are from the L_2

optimization, and all images representing L_2 are also representative of our L_{inf} results. The accepted threshold for an imperceptible attack on CIFAR10 under the L_2 setting is: $L_2 = 0.5$.⁷ The perturbations of both the GTS Attack_w and GTS Attack_s are nearly three times lower than this threshold for attacks against the vanilla model, and two times smaller on adversarial training (AT) and gradient regularization (GR) defenses. Notably, these attacks demonstrate significant effectiveness against the TRADES defense. GTS Attack shows similar performance to CW attack and significant

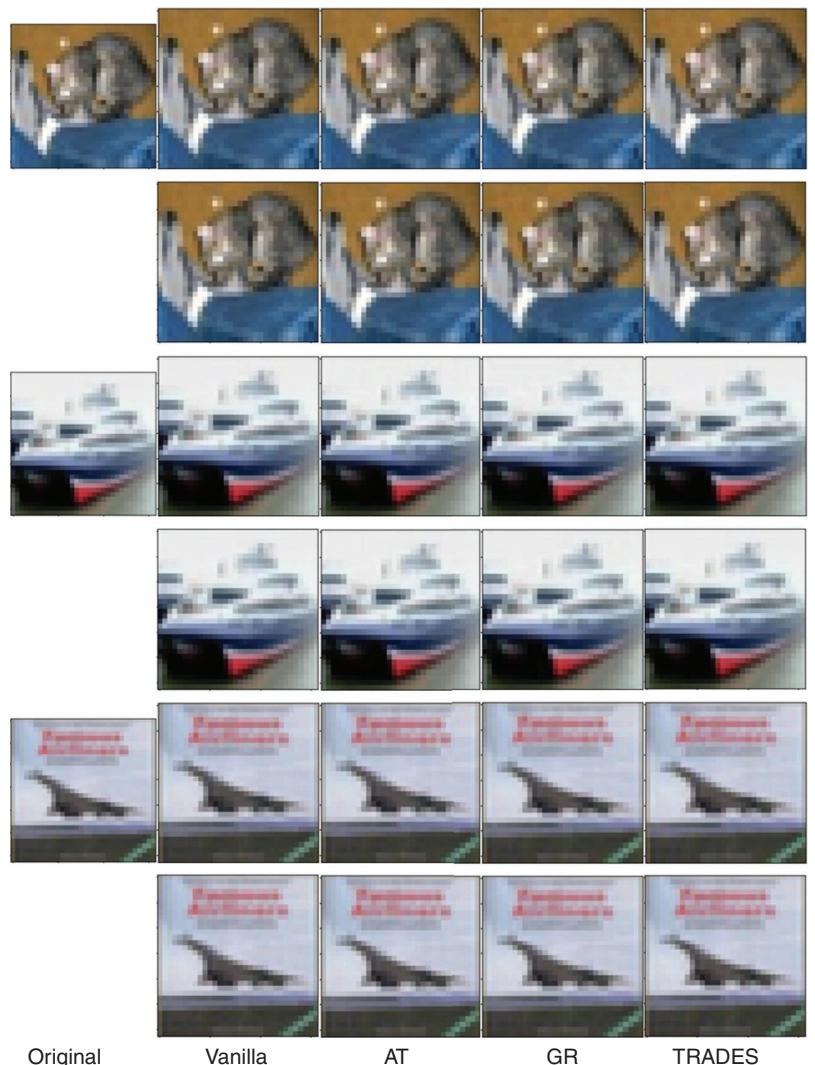


FIGURE 4. GTS Attacks optimized for L_2 with CIFAR10 on all four models. The top row of each image is the GTS Attack_w and the next row of the same image in the GTS Attack_s. These images were the first three images in the CIFAR10 test set.

TABLE 3. CIFAR10: Targeted attack results.

Vanilla				AT		GR		TRADES	
mean	ASR	mean	ASR	mean	ASR	mean	ASR	mean	ASR
L_2	A* Attack _w	0.27	100%	0.33	96%	0.34	100%	0.21	99%
	A* Attack _s	0.25	97%	0.42	98%	0.32	100%	0.24	100%
	CW	0.27	84%	0.37	77%	0.27	94%	0.26	90%
	PGD	0.39	88%	0.40	76%	0.40	83%	0.30	77%
	AutoAttack	0.50	90%	0.50	81%	0.50	88%	0.50	85%
L_∞	A* Attack _w	0.0226	100%	0.0331	97%	0.0213	100%	0.0077	99%
	A* Attack _s	0.0461	99%	0.0529	98%	0.0357	100%	0.0290	100%
	CW	0.0116	90%	0.0137	90%	0.024	95%	0.0122	98%
	PGD	0.0196	85%	0.0198	89%	0.0200	95%	0.0197	98%
	AutoAttack	0.0100	82%	0.0500	99%	0.0500	100%	0.0500	100%
L_0	A* Attack _w	5.28	98%	6.84	95%	6.45	98%	4.70	100%
	A* Attack _s	5.07	98%	7.27	97%	6.71	100%	4.60	100%

TABLE 4. CIFAR10: Untargeted black-box hybrid attack results.

Vanilla				AT				GR				TRADES			
		mean	Queries	ASR	mean	Queries	ASR	mean	Queries	ASR	mean	Queries	ASR	mean	Queries
$L_2 A^*$	Hybrid _w	0.44	14	97%	0.62	19	93%	0.61	20	99%	0.71	18	93%		
	A* Hybrid _s	0.45	14	98%	0.62	18	95%	0.62	19	100%	0.72	16	98%		
	HSJA	3.56	57	100%	7.90	53	94%	8.34	52	98%	5.56	51	93%		
	CW	0.48	1	66%	0.81	1	64%	0.77	1	54%	0.71	1	58%		
	PGD	0.30	1	45%	0.50	1	38%	0.50	1	48%	0.50	1	17%		
	AutoAttack	0.30	1	48%	0.70	1	55%	0.70	1	64%	0.50	1	47%		
$L_\infty A^*$	Hybrid _w	0.0151	5	98%	0.0426	23	99%	0.0291	24	99%	0.0375	2	100%		
	A* Hybrid _s	0.0381	14	98%	0.0552	18	95%	0.0540	20	100%	0.0699	17	98%		
	HSJA	0.0700	68	99%	0.1800	65	96%	0.2000	64	96%	0.1300	63	92%		
	ECO	0.0431	62	72%	0.0431	62	66%	0.0378	58	63%	0.0347	54	58%		
	CW	0.0215	1	49%	0.0368	1	55%	0.0251	1	42%	0.0367	1	59%		
	PGD	0.0500	1	32%	0.0693	1	34%	0.0692	1	44%	0.0682	1	39%		
	AutoAttack	0.0200	1	82%	0.0400	1	86%	0.0300	1	82%	0.0400	1	87%		
$L_0 A^*$	Hybrid _w	7.42	18	100%	12.47	34	100%	10.97	26	100%	8.64	21	99%		
	A* Hybrid _s	8.9	22	100%	12.12	32	100%	10.93	26	100%	8.85	20	99%		

improvement compared to PGD and AutoAttack.

Similarly, the conventional threshold for L_{inf} on CIFAR10 is $L_{\text{inf}} = 0.031$. The results from GTS Attack_w significantly surpassed this threshold achieving values seven times lower against the vanilla model. Even against the robust TRADES defense, the values were more than three times lower, all while sustaining a 100% attack success rate. Although GTS Attack_w outperformed GTS Attack_s in L_{inf} , GTS Attack_s still yielded competitive results, matching or even surpassing the predefined threshold, all while upholding an ASR of 100%. Figure 4 presents sample images from the L_2 optimized attack for the first three unique classes in our CIFAR10 test set, applied to our models. Notably, in each instance, the adversarial images are indistinguishable from the original image. For the L_0 distance metric, our attack finds adversarial examples with less than four changed pixels on average for the vanilla model and fewer than six changed pixels in every case for both attacks as seen in Table 2.

Targeted attacks

The GTS attack produced competitive white-box and score-based targeted attack results. Table 3 shows the results of our targeted GTS Attack_w and GTS Attack_s. In each case, GTS Attack has a mean L_2 value below 0.5 with similar values compared to CW, and much better success when compared with PGD and AutoAttack. The CW attack performed better under the L_{inf} optimization for but at the cost of a lower ASR.

The L_0 optimized results are analogous across all model types to those reported for the CW attack under similar settings for their undefended model.⁵ They report an average L_0 value of 5.9 with an ASR of 100%. Both GTS Attack_w and GTS Attack_s have similar distortion with an ASR of 98% on the vanilla model while maintaining comparable results on the defended models as seen in Table 2.

GTS Black-Box Hybrid Attacks

The GTS Black-Box Hybrid Attacks were conducted using both the GTS Attack_w and GTS Attack_s as the substitute model. We denote this distinction as GTS Hybrid_w and GTS Hybrid_s, respectively. Table 4 shows the GTS Attack achieving an ASR of 93% or greater on every attack with an average model query count of 24 or less. Notably, on average the L_2 vanilla model attacks were below the standard 0.5 threshold, with the defended models having slightly more perturbation. By

comparison, the decision-based HSJA has significantly more perturbation at more than double the number of model queries, while the ECO attack under L_{inf} shows comparable perturbation to the GTS Attacks, it requires more than 12 times as many model queries as GTS Hybrid_w at a much lower ASR. At comparable L_2 distortions with the GTS Attacks, the transfer-based attacks have significantly lower ASRs. When comparing L_{inf} results, AutoAttack performs very well as a transfer-based attack, but is still outperformed by

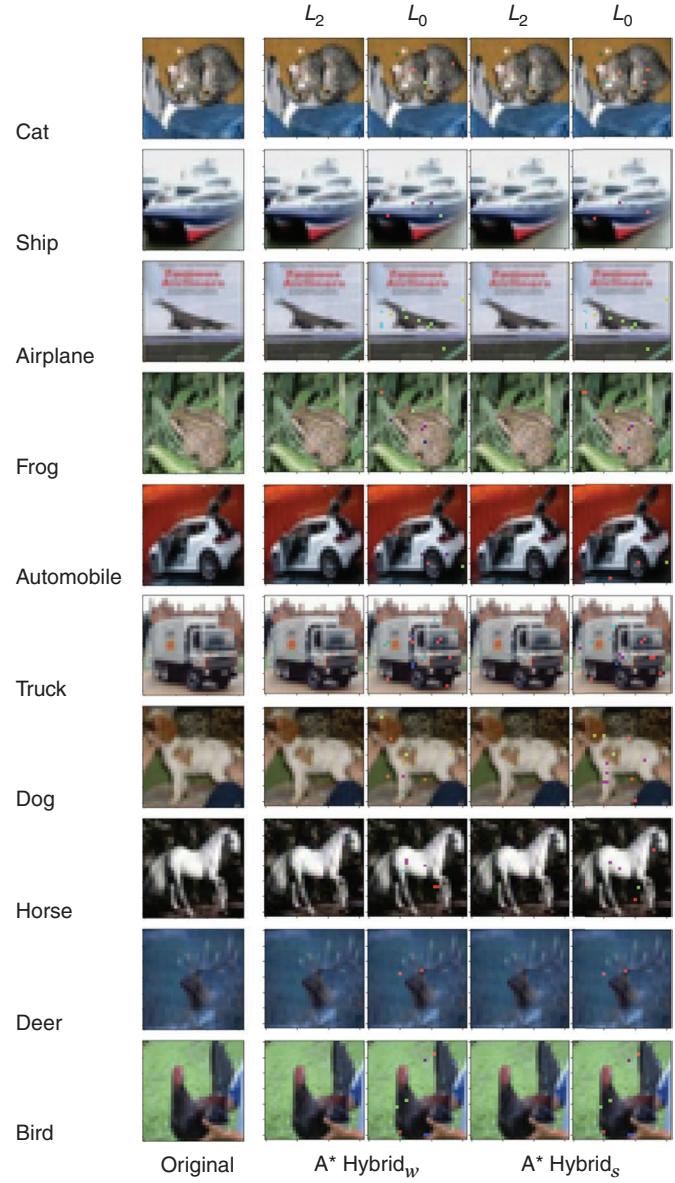


FIGURE 5. GTS Black-Box Hybrid Attacks on CIFAR10 for L_2 and L_0 optimizations.

GTS Attack when comparing ASRs. It is worth noting that AutoAttack's results at the same L_{inf} value as the GTS Attack have much more obvious perturbation. Figure 5 shows the first 10 classes as they appear in the CIFAR10 test dataset for both GTS Hybrid attacks. The L_2 optimized images are indistinguishable from the original images, while the L_0 optimized images have very few noticeable pixel perturbations.

This article introduced the GTS Attack, a suite of diverse adversarial example attacks that leverage the GTS algorithm. Our findings underscore GTS Attack's efficiency, particularly in the black-box setting, outperforming existing attacks with fewer model queries and improved success rates. This shows the need for continuous evaluation of model robustness, especially as MLaaS becomes more prevalent. The GTS Attack's success against state-of-the-art defenses highlights potential vulnerabilities. While this research explores a new approach to the development of adversarial examples, it also underscores the dynamic nature of adversarial machine learning. As we push the boundaries with attacks like GTS Attack, it prompts the development of more resilient defense mechanisms. ■

REFERENCES

1. M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein, "Square attack: A query-efficient black-box adversarial attack via random search," 2020, *arXiv:1912.00049*.
2. J. A. Baktash and M. Dawodi, "Gpt-4: A review on advancements and opportunities in natural language processing," 2023, *arXiv:2305.03195*.
3. A. Betti, M. Gori, and S. Melacci, *Deep Learning to See: Towards New Foundations of Computer Vision* (SpringerBriefs in Computer Science), 1st ed. Cham, Switzerland: Springer-Verlag, 2022.
4. W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," 2018, *arXiv:1712.04248*.
5. N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," 2017, *arXiv:1608.04644*.
6. J. Chen, M. I. Jordan, and M. J. Wainwright, "HopSkipJumpAttack: A query-efficient decision-based attack," 2020, *arXiv:1904.02144*.
7. F. Croce et al., "RobustBench: A standardized adversarial robustness benchmark," 2020, *arXiv:2010.09670*.
8. F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," 2020, *arXiv:2003.01690*.
9. Y. Dong, S. Cheng, T. Pang, H. Su, and J. Zhu, "Query-efficient black-box adversarial attacks guided by a transfer-based prior," 2022, *arXiv:2203.06560*.
10. I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2015, *arXiv:1412.6572*.
11. A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2019, *arXiv:1706.06083*.
12. R. Miotto et al., "Deep learning for healthcare: Review, opportunities and challenges," *Briefings Bioinf.*, vol. 19, no. 6, pp. 1236–1246, Nov. 2018, doi: [10.1093/bib/bbx044](https://doi.org/10.1093/bib/bbx044).
13. S. Moon, G. An, and H. O. Song, "Parsimonious black-box adversarial attacks via efficient combinatorial optimization," 2022, *arXiv:1905.06635*.
14. N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: From phenomena to black-box attacks using adversarial samples," 2016, *arXiv:1605.07277*.
15. A. S. Ross and F. Doshi-Velez, "Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients," 2017, *arXiv:1711.09404*.
16. S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. New York, NY, USA: Pearson, 2018.
17. C. Szegedy et al., "Intriguing properties of neural networks," 2014, *arXiv:1312.6199*.
18. F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," 2020, *arXiv:1705.07204*.
19. C. Wilt, J. Thayer, and W. Ruml, "A comparison of greedy search algorithms," in *Proc. Int. Symp. Combinatorial Search*, 2010, vol. 1, no. 1, pp. 129–136, doi: [10.1609/socs.v1i1.18182](https://doi.org/10.1609/socs.v1i1.18182).
20. H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. E. Ghaoui, and M. I. Jordan, "Theoretically principled trade-off between robustness and accuracy," 2019, *arXiv:1901.08573*.
21. H. Zhu, S. Zhang, and K. Chen, "AI-guardian: Defeating adversarial attacks using backdoors," in *Proc. IEEE Symp. Secur. Privacy (SP)*, 2023, pp. 701–718, doi: [10.1109/SP46215.2023.10179473](https://doi.org/10.1109/SP46215.2023.10179473).
22. M. Akers and A. Barton, "Forming adversarial example attacks against deep neural networks with reinforcement learning," *Computer*, vol. 57, no. 1, pp. 88–99, Jan. 2024, doi: [10.1109/MC.2023.3324751](https://doi.org/10.1109/MC.2023.3324751).

CHRISTOPHER CLARK is a communications officer in the United States Marine Corps. He is currently the Artificial Intelligence Lead for the Deputy Commandant for Information at the Pentagon, Washington, DC, VA 20310 USA. Contact him at chris.d.clark5@gmail.com.

ARMON BARTON is an assistant professor in the Department of Computer Science at Naval Postgraduate School, Monterey, CA 93943 USA. Contact him at armon.barton@nps.edu.



Software Testing in the Generative AI Era: A Practitioner's Playbook

Douglas C. Schmidt^{ID}, William & Mary

This article presents a practitioner-focused playbook on how generative artificial intelligence is transforming software testing by automating tasks like test generation, simulation, and anomaly detection, while also introducing new challenges like hallucinations, bias, and nondeterminism.

Software testing is undergoing a paradigm shift due to generative artificial intelligence (AI). Imagine test cases that write themselves by analyzing your documentation and code, user behaviors being simulated before any real user ever touches the software, and continuous



COPYRIGHT ISTOCKPHOTO, CREDIT: VANDY

feedback guiding development in real time. Generative AI—especially large language models (LLMs) trained on vast data sets—is turning this vision into reality. These tools can analyze your requirements and code to produce test cases automatically, synthesize realistic test scenarios on the fly, and even monitor running systems for anomalies. The promise is compelling: you and your team can test faster, smarter, and more thoroughly.

However, these AI-augmented advances also introduce new risks. The same AI tools that boost testing can produce unpredictable behavior—from biased outputs or “hallucinated” results to inconsistent, nondeterministic responses that are hard to reproduce or verify. In high-stakes domains like health care, finance, and defense, where lives and missions are on the line, ensuring the reliability and trustworthiness

of AI-augmented testing is paramount. Traditional quality assurance (QA) practices must adapt to issues like AI bias, unexpected outputs, and regulatory compliance.

This article explores how generative AI is reshaping software testing—the benefits it offers, the challenges it raises, and how you can adapt. With AI coding assistants like OpenAI's Codex, GitHub Copilot, and Meta's Code Llama increasingly integrated development environments, continuous integration/continuous delivery (CI/CD) pipelines, and cloud platforms, teams need to adjust quickly. Understanding how these tools work—and recognizing when they don't—is becoming a critical skill. The guidance here draws on my recent stint as the Director of Operational Test and Evaluation and related collaborations with colleagues.^{1,2,3}

HOW GENERATIVE AI IS AUTOMATING AND SCALING TESTING TODAY

Generative AI is extending the reach of test automation. Instead of manually writing every test or trying to guess all possible user behaviors, you can offload many tedious and error-prone tasks to AI. The AI-augmented testing

capabilities shown in [Figure 1](#) serve as accelerators for software QA, and are described as follows.

Automatic test case generation

Tools like Diffblue Cover and CodiumAI can analyze your code and generate unit tests across a variety of languages. They can even integrate with your continuous integration pipelines (for example, Jenkins or GitLab) for automatic execution.⁴ For instance, an LLM can read a specification and produce a suite of tests covering the expected behaviors. This automation helps validate intended functionality early and targets known weak spots. It also expands coverage while reducing the manual effort of writing tests.

Continuous testing and feedback

Generative AI supports continuous testing by monitoring systems and providing rapid feedback. Machine learning models can watch test executions or even live production logs to spot anomalies and failures in real time.⁶ If a test fails, an AI assistant might immediately analyze the recent code changes and suggest likely causes. AI can also detect patterns in past defects to recommend new test cases or fixes. In modern

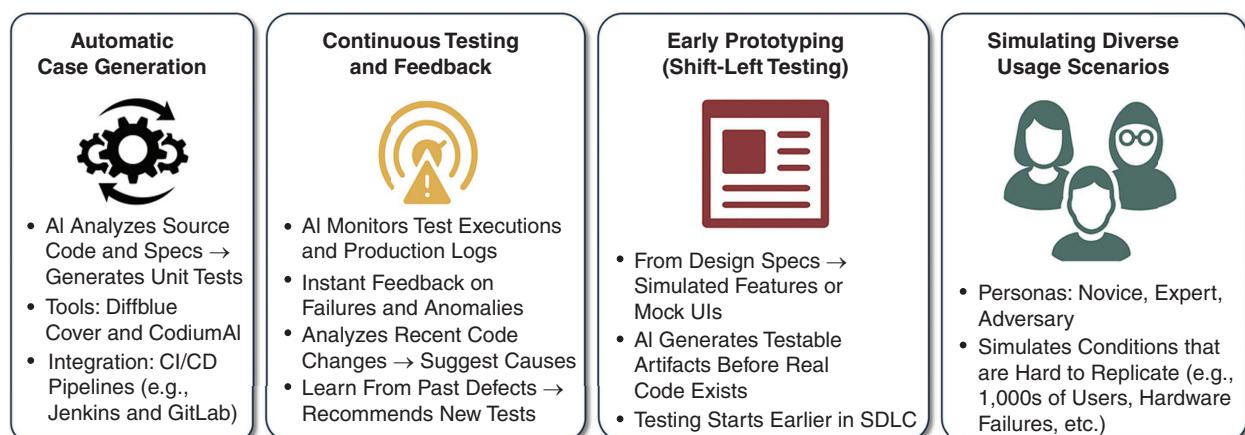
Development, Operations, and Security pipelines, every code commit can trigger AI-generated tests and anomaly detection, providing instant feedback and catching issues earlier.

Early prototyping and “shift-left” testing

Generative AI blurs the line between development and testing by enabling earlier prototyping of features. Rather than waiting for a fully integrated system to begin testing, an AI tool can create a mock interface or simulate a new feature from just a design description. This “shift-left” approach allows you to exercise and evaluate components before writing any real code, thus shortening the QA feedback loop and catching design flaws sooner in the development process.

Simulating diverse usage

Generative AI enables you to simulate a wide range of user behaviors and environmental conditions. For instance, you might prompt an LLM to act as various personas⁵ (for example, a novice user, an expert, even an adversarial attacker) to see how your software handles different inputs and sequences of actions. AI-augmented simulations can also mimic conditions that are



AI-Augmented Testing = Higher Test Coverage + Faster Release + Fewer Defects

AI doesn't just Automate — it Amplifies QA Teams to Deliver Reliable Software Faster

FIGURE 1. Accelerating software testing with generative AI. UI: user Interface; SDLC: software development life cycle.

impractical to reproduce manually (imagine thousands of virtual users or sudden sensor failures). Testing under these diverse scenarios gives you greater confidence in the robustness of your system.

These AI-augmented techniques can widen your test coverage (including edge cases), catch bugs earlier, and speed up the feedback cycle. By offloading tedious work to AI and injecting intelligence into testing, generative AI becomes a force multiplier for QA, enabling you to deliver more reliable software faster.

CHALLENGES OF AI-AUGMENTED TESTING

Along with the benefits of AI-augmented testing come new challenges, there are several key areas you need to address, which are shown in Figure 2 and described as follows.

AI “hallucinations”

AI systems sometimes produce answers or code that look correct and confident but are completely wrong.⁷ For example, an AI-generated test might assert an output value that appears plausible (perhaps it even aligns with the documentation) but is incorrect. If you rely on that test, it could pass or fail for the wrong reasons. In short, you can't blindly trust AI-generated outputs—especially for critical logic—with verification. Keep a human in the loop to review important AI-generated tests and results.⁸ Likewise, perform adversarial testing; for example, feed malformed or malicious inputs to the AI and see if the system goes off track. Your goal is to ensure the system handles uncertainty properly, for example, by indicating when it doesn't know an answer instead of confidently fabricating one.

Nondeterministic behavior

Unlike traditional software, AI components might not behave the same way every time. A conventional function will produce identical outputs for identical inputs, but an AI model might respond slightly differently on each run,

which makes it hard to consistently reproduce results. For example, if a test passes once but fails on the next run, is that due to randomness or a real bug? You may need to rerun certain tests numerous times to understand the range of possible outcomes. Even small input tweaks or data updates

shouldn't, such as make an unauthorized decision or reveal sensitive information. Emerging regulations like the European Union's AI Act will likely mandate evidence of an AI system's fairness, transparency, and risk controls. AI tools can assist by scanning for compliance issues,⁹ but ultimately,

Generative AI blurs the line between development and testing by enabling earlier prototyping of features.

can shift an AI's responses, so watch out for model drift—when an AI's behavior changes over time as its data or parameters evolve. Ensuring quality for nondeterministic systems means gathering far more test data and often defining new metrics. It's a shift away from the traditional “pass/fail” mindset toward measuring whether the AI's performance stays within an acceptable range of reliability.

Compliance and ethics

In regulated domains, you must ensure that AI systems comply with all relevant laws and ethical standards, for example, a healthcare AI should never leak patient data. Your testing should include scenario-based checks that exercise policy constraints, for example, check that AI never does something it

you and your team are accountable for ensuring that AI-augmented software stays within legal and ethical bounds.

Bias and fairness

Generative models learn from their training data, so any biases in that data can creep into AI-generated tests or outputs. If the training data underrepresents certain user groups or conditions, the AI might overlook those scenarios, leading to blind spots in quality. So don't just ask “Does it work?”—also ask “Does it work for everyone using our software?” In practice, you'll need to use diverse inputs, cover underrepresented scenarios, and watch for skewed or unfair results. If you detect bias, you may need to adjust your test data or retrain the model to improve its fairness, even if

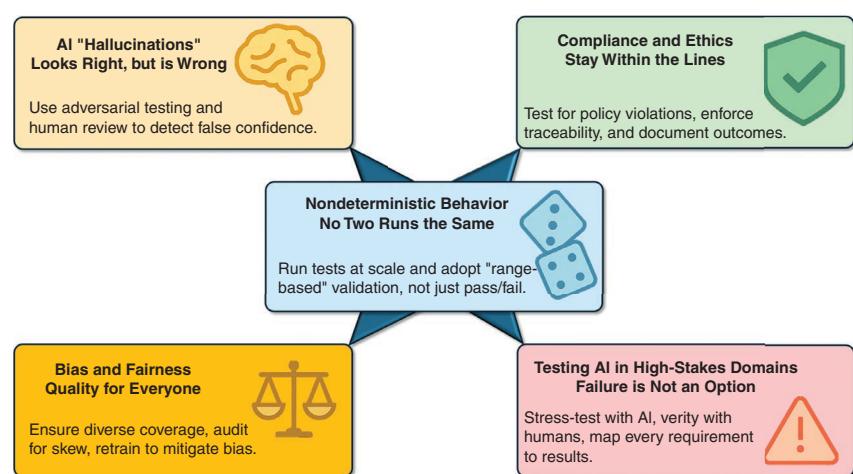


FIGURE 2. Challenges of AI-augmented software testing.

that takes extra effort. These considerations broaden the notion of software QA to include factors like fairness, explainability, robustness under uncertainty, and ethical integrity. They aren't insurmountable but addressing them requires evolving your testing strategies—especially in safety-critical environments where even a small AI misstep can be disastrous.

Testing AI in high-stakes domains

The stakes are elevated in high-stakes domains like defense, aerospace, and health care, which can't tolerate failures in AI-augmented systems. A flaw in an autonomous emergency response drone's software or a medical decision-support AI could be deadly. In these cases, you must demonstrate that the system works correctly even under extreme or unusual conditions. Generative AI can help by simulating numerous variations of scenarios (for example, different operating conditions, emergency situations, rare edge cases) to stress-test the system and expose weaknesses.

Even so, rigorous human oversight is essential, so “trust but verify” at every step. If AI-augmented testing tools report everything passed, you should still spot-check the results to identify edge cases the AI might have missed. In high-stakes

domains, even rare oversights can be unacceptable. Traceability is also crucial, so document how each requirement maps to specific tests/outcomes to justify your test coverage and explain test results to stakeholders and regulators. Whenever AI is involved in a regulated, safety-critical system, be prepared to show exactly what was tested and verify nothing was left to chance.

The lesson from high-stakes domains is clear: use AI to amplify your testing, but maintain strict human oversight and accountability. Reliability, transparency, and safety must remain paramount whenever AI is part of a mission-critical system.

ADAPTING TESTING PRACTICES FOR THE AI ERA

To harness generative AI in testing while mitigating its risks, you'll need to evolve your QA practices. Several ways to update your approach for the AI era are shown in [Figure 3](#) and described as follows.

Foster AI literacy in QA teams

Ensure your testing team learns how AI works—and where it can fail. Testers should learn the basics of model training and evaluation, familiarize themselves with common AI failure modes, and adopt prompt engineering

techniques and patterns⁵ to apply LLMs effectively. This knowledge helps them use AI-based tools robustly and recognize when an AI's output might be misleading. Encourage your team to practice crafting prompts and critically analyze AI-generated results. With greater AI literacy, your team can treat AI as a powerful assistant under their supervision, rather than a mysterious black box.

Encourage cross-functional collaboration

Testing AI-infused systems isn't just the QA team's job. Developers, testers, data scientists, and domain experts should all collaborate on AI-augmented projects. Involve QA early in the design of AI features so they can advise on testability and anticipate problems. Likewise, developers should build transparency and test hooks into AI components, for example, logging an AI model's key decision factors for later review. Involve domain experts to ensure test scenarios reflect real-world conditions, for example, have medical doctors contribute cases to test a healthcare AI or have financial experts suggest scenarios to test fintech AI. Breaking down silos and sharing knowledge across roles yields more robust testing strategies for AI-augmented systems.



Foster AI Literacy in QA Teams
Understand AI Concepts and Limitations



Encourage Cross-Functional Collaborations
Collaborate With Developers and Domain Experts



Upgrade Your Toolkit and Techniques
Augment Testing With AI-Driven Tools



Adopt Continuous Monitoring and Feedback
Track AI System Performance in Production

Upgrade your tool kit and techniques

Keep your proven QA methods but augment them with AI-augmented approaches. For example, introduce adversarial testing (to try “breaking” the AI with malicious or unexpected inputs) and apply statistical analysis to interpret test results since AI outputs are probabilistic rather than deterministic.³ You may need new metrics to quantify qualities like bias or uncertainty in the AI's output. Likewise, experiment with emerging AI-augmented test generation and analysis tools to see how they can boost your productivity. By expanding your toolbox, you can better cover AI-specific

FIGURE 3. Adapting testing practices for the AI era.

failure modes and make your testing regime more effective.

Adopt continuous monitoring and feedback

Don't treat testing as a one-and-done phase. After deployment, continue to monitor your AI components in production and feed those insights back into your tests. AI-augmented monitoring tools (such as Amazon DevOps Guru or Sentry) can watch production systems for anomalies, performance drift, or new types of errors, then alert your team. If a model's performance degrades over time, be ready to update test cases or retrain the model as needed. Schedule regular audits of AI outputs to catch issues that only appear with production deployments. AI can automate parts of this monitoring and analysis, but human experts should review the results and guide improvements. Maintaining this feedback loop will keep your AI systems reliable long after deployment.¹⁰

By putting these strategies into practice, you can turn generative AI into a real asset in your QA process. Testing will become more adaptive and comprehensive, keeping pace with AI-augmented development while upholding high standards of quality and trust.

YOUR AI TESTING PLAYBOOK: WHAT TO DO NEXT

Generative AI is rapidly reshaping software testing and early adopters are already seeing significant benefits. Autogenerated tests and large-scale simulations are catching bugs that previously slipped through manual efforts and faster feedback loops are helping teams iterate quickly. When applied properly, these techniques can yield more reliable software and shorter development cycles by enabling smarter testing.

At the same time, you need a clear view of AI's limitations. An AI-augmented system is only as good as the data and algorithms behind it, so it can still exhibit bias, produce incorrect results, or behave unpredictably.

Human judgment, disciplined testing practices, and strong processes remain essential to validate and rein in AI contributions. Remember to "test the tester" by building in safeguards to catch mistakes or quirks in the AI tools themselves. By updating your practices and skill sets to account for these issues, you can harness the benefits of AI while mitigating its risks.

So what should you do now? Figure 4 shows a few concrete next steps for different roles on your team, which are explained as follows.

› **Software engineers and developers:** Integrate generative AI tools into your development and testing workflow to boost coverage and catch issues early. Use AI to generate unit tests and to simulate complex user flows. Don't assume the AI is always right—review and refine every AI-generated test case or code suggestion. Design your software with testability in mind, for example, add logging and hooks around AI-augmented features so you can verify their outputs. Let AI handle the tedious parts of testing but apply your judgment to discern what's correct.

› **QA and testing professionals:** Leverage AI as a partner to extend your testing. Use AI tools to generate diverse test scenarios and synthetic data, uncovering bugs that traditional methods might miss. Focus on validating the AI's behavior, for example, look for biased outputs or hallucinations and apply adversarial testing to stress-test AI-augmented features. Champion new types of checks in your test plans (for example, include explicit fairness and bias tests for AI outputs). Orchestrate the best of both worlds—use AI to work more efficiently while you focus on ensuring accuracy and integrity.

› **Technical leaders:** Create an environment that integrates AI into quality assurance thoughtfully. Invest in training your teams on AI tools and concepts and give them room to experiment with AI-augmented testing solutions. Update your QA processes to add checkpoints for AI-related risks, for example, require bias testing for machine learning components, and mandate human review of critical AI-augmented decisions or recommendations. Promote cross-team collaboration so that development, QA,

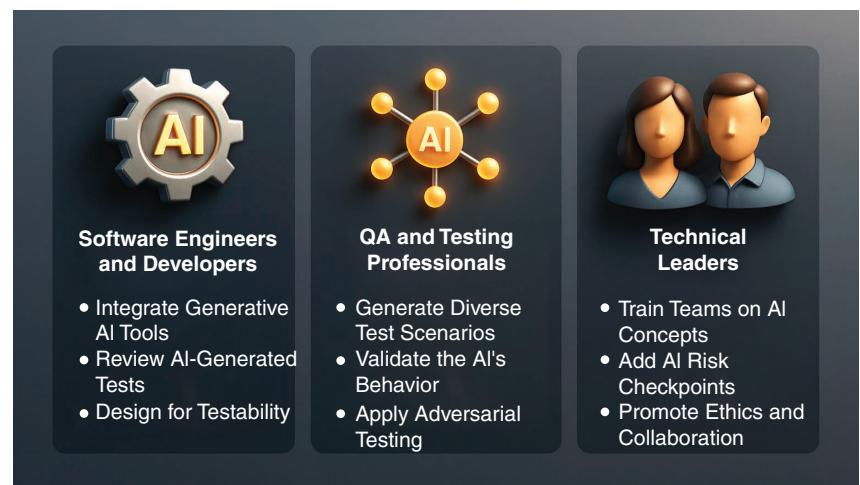


FIGURE 4. Testing roles for AI-augmented software systems.

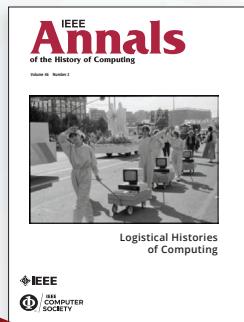
and data science teams are aligned on quality goals. Likewise, insist on high ethical standards (such as transparency, fairness, and accountability) for any AI usage in your products and processes. With strong leadership and the right guardrails, your organization can harness generative AI responsibly to deliver better software faster! □

REFERENCES

1. L. J. Freeman, D. C. Schmidt, A. Bonnell, J. Robert, H. Wojton, and Acquisition Innovation Research Council, AIRC Panel on Generative AI in the Acquisition Lifecycle. (Jan. 7, 2025). [Online Video]. Available: <https://www.youtube.com/watch?v=ZlCc94w-2bY>
2. L. J. Freeman, J. Robert, and H. Wojton, "The impact of generative AI on test & evaluation: Challenges and opportunities," in Proc. Int. Workshop Envisioning AI-Augmented Softw. Develop. Lifecycle, Trondheim, Norway, 2025, pp. 1–5.
3. P. Roy, J. Chandrasekaran, E. Lanus, L. J. Freeman, and J. Werner, "A survey of data security: Practices from cybersecurity and challenges of machine learning," 2023, *arXiv:2310.04513*.
4. F. Ricca, A. Marchetto, and A. Stocco, "AI-based test automation: A grey literature analysis," in Proc. IEEE Int. Conf. Softw. Testing, Verification Validation Workshops (ICSTW), 2021, pp. 263–270, doi: [10.1109/ICSTW52544.2021.00051](https://doi.org/10.1109/ICSTW52544.2021.00051).
5. J. White et al., "A prompt pattern catalog to enhance prompt engineering with ChatGPT," in Proc. 30th Pattern Lang. Program. (PLoP) Conf., Allerton Park, IL, USA, pp. 1–31, 2023.
6. P. Dogga, K. Narasimhan, A. Sivaraman, S. Saini, G. Varghese, and R. Netravali, "Revelio: ML-generated debugging queries for finding root causes in distributed systems," in Proc. Mach. Learn. Syst. 4 (MLSys), pp. 601–622, 2022.
7. F. Liu et al., "Exploring and evaluating hallucinations in LLM-powered code generation," 2024, *arXiv:2404.00971*.
8. N. Maleki, B. Padmanabhan, and K. Dutta, "AI hallucinations: A misnomer worth clarifying," in Proc. IEEE Conf. Artif. Intell. (CAI), 2024, pp. 133–138, doi: [10.1109/CAI59869.2024.00033](https://doi.org/10.1109/CAI59869.2024.00033).
9. A. S. Chandrasekaran, "Harnessing the power of generative artificial intelligence (GenAI) in governance, risk management, and compliance (GRC)," *Int. Res. J. Eng. Sci. Technol. Innov.*, vol. 11, no. 5, p. 11, 2024.
10. E. Lanus, B. J. Lee, L. Pol, D. Sobien, J. Kauffman, and L. J. Freeman, "Coverage for identifying critical metadata in machine learning operating envelopes," in Proc. IEEE Int. Conf. Softw. Testing, Verification Validation Workshops (ICSTW), 2024, pp. 217–226.

DOUGLAS C. SCHMIDT is dean of the School of Computing, Data Sciences & Physics, William & Mary, Williamsburg, VA 23185 USA. Contact him at douglas.c.schmidt@wm.edu.

IEEE Annals of the History of Computing



IEEE Annals of the History of Computing publishes work covering the broad history of computer technology, including technical, economic, political, social, cultural, institutional, and material aspects of computing. Featuring scholarly articles by historians, computer scientists, and interdisciplinary scholars in fields such as media studies and science and technology studies, as well as firsthand accounts, *Annals* is the primary scholarly publication for recording, analyzing, and debating the history of computing.



www.computer.org/annals



Digital Object Identifier 10.1109/MC.2025.3573140



Generative AI and Health Care: Brief Survey

Roy Rada^{ID}, University of Maryland

AI is transforming health care. Big Data radiation oncology and high-touch palliative care illustrate different opportunities.

Artificial intelligence (AI) and health care have impacted one another since the beginning of AI. Traditionally, the branches of health care most dependent on digitized information benefited the most from AI. Do the latest advances in natural language processing impact the trajectory? Since large language models (LLMs) exploit free-form text, do they open opportunities for branches of health care traditionally reliant on human-human interaction? Or will the areas that get the most investment already be the ones that get the most investment in the future, as a continuation of the famous aphorism “the rich get richer, while the poor get poorer.”

The history of computer applications in health care supports the famous “Rich get Richer” phenomenon. In 1968,

Merton introduced the “Matthew effect”¹ as “This complex pattern of the misallocation of credit for scientific work must quite evidently be described as the Matthew effect ... For unto everyone that hath shall be given.” Evidence for the Matthew effect has appeared across occupations. A study of health care delivery in China concludes, “the Inter-city Health Network generates new

structural inequalities in health care access exhibiting a Matthew effect.”² New investments in digital information systems in health care have typically gone into areas that most relied on digital information.³ Historically, the first health care computer investments were in accounting and next, radiology and pathology. Health care and AI illustrate the Matthew effect. This essay next reviews the AI and health literature before focusing on two application areas, radiation oncology and palliative care, which sit at opposite extremes of the capital spectrum.

AI AND HEALTH CARE

Deep learning has been successfully applied to clinical medicine for more than three decades.⁴ Generative AI trains on multimedia libraries, while LLMs train on document libraries,⁵ and both perform well on many health-care tasks.⁶ While LLMs lack epistemic validity,⁷ they

create new opportunities for health care applications that converse with humans

A search of IEEE Xplore for [“artificial intelligence” AND (“health care” OR medicine) AND “systematic reviews” AND 2020-to-2025] returned 54 citations. Most citations were for highly focused topics, such as “multi-modal deep learning diagnosis of Parkinson’s Disease,”⁸ but nine were broad. One broad review shows that Big Data facilitates personalized medicine, predictive analytics, early illness diagnosis, precision diagnostics, and treatment optimization from electronic health records, genomes,

occurs in real time under computer control. Radiation oncology relies on massive amounts of digital information and repetitive tasks¹³—“Big Data” incarnate.¹⁴

Radiation oncology is ripe for AI applications¹⁵: “AI … transformative applications in radiation oncology given … a heavy reliance on digital data processing and computer software.” Radiation oncologists appreciate AI’s impact and say that AI revolutionizes radiation therapy.¹⁶ AI is transformative at every step of the radiation oncology process.

Palliative medicine differs sharply from radiation oncology. The World

NIH-funded research projects and access patents resulting from NIH funding.²¹ A search on this NIH repository for “radiation oncology” AND “artificial intelligence” returns 68 active projects linked to 60 patents. A search on “palliative care” AND “artificial intelligence” retrieves 10 active projects and 0 patents. Radiation oncology is “richer” than palliative medicine.

How are LLMs special for palliative care? LLMs can forecast mortality or monitor pain. For predicting mortality, one project uses traditional machine learning on structured data sets,²² while another project uses an LLM that reads unstructured medical records.²³ These two projects highlight the salient difference between LLM and non-LLM approaches. The LLM approach works on unstructured natural language. The LLM approach lends itself more readily to extensions and diffusion into electronic health records systems.

An LLM for patient mental health showed that the LLM system responds like a professional mental health counselor.²⁴ For the doctor, an LLM inferred wishes of mentally incapacitated patients based on prior records.²⁵ LLMs are well suited to palliative care.²⁶

WORKFLOW

One challenge in the roll out of LLM applications is gaining the trust of the intended users. While this trust is a necessary condition for diffusion, another necessary condition is fitting into the workflow. The surge of interest in expert systems in the 1980s bares comparison to the surge of interest in LLMs. The 1980s MYCIN expert system was as good as expert physicians in its problem area of bacteremia, and it offered recency, accuracy, coherence, and transparency confirmed by clinical trials.²⁷ However, to use MYCIN, the physician needed to access a separate, stand-alone application and reenter patient information into that application which took enough time that the cost/benefit ratio was too high for the physician—in other words, MYCIN did

Large companies that develop electronic health records systems are working with large provider networks to integrate LLMs into the workflow.

wearables, and medical imaging.⁹ In another broad review, adoption of AI applications depends on the factors of technical feasibility, ease of use, system quality, performance, usability, social influence, and trust.¹⁰ The IEEE literature suggests that AI increases the effectiveness and efficiency of health care.¹¹

RADIATION ONCOLOGY AND PALLIATIVE CARE

Radiation oncology is one of three oncology branches, the other two being medical and surgical oncology. In preparing a radiotherapy treatment plan, the doctors acquire images, segment the target, and plan doses to kill cancer cells. Radiomics is a method to quantitatively analyze medical images to uncover tumoral patterns not appreciated by the naked eye and to support personalized therapy.¹² Doctors try to predict outcomes based on both radiomics and genomics. In the delivery of radiation, the team monitors patient changes and adapts treatment. Changing radiation based on patient motion

Health Organization states¹⁷: “Palliative care … improves the quality of life of patients and their families who are facing … life-threatening illness, through the prevention and relief of suffering … whether physical, psychosocial, or spiritual.” Diseases benefiting from palliative medicine include cancer, heart failure, and stroke. Nevertheless, only 14% of people who need palliative care currently get it.¹⁷ Health insurance underfunds palliative care, and hospitals, at a loss to themselves, provide more than half of the overall cost of palliative care.¹⁸ A systematic review of challenges in palliative care concludes that the most significant barrier is a lack of resources.¹⁹ If technology could reduce costs of delivering palliative care, then palliative care professionals have an obligation to explore using it.²⁰

Palliative care uses AI less than radiation oncology. Grant and patent activity evidences the divide. With the U.S. National Institutes of Health (NIH) “Research Portfolio Online Reporting Tools,” users search a repository of both intramural and extramural

not fit into the workflow.³ Large companies that develop electronic health records systems are working with large provider networks to integrate LLMs into the workflow.²⁸ However, journals underreport these efforts.

Workflow systems rely on roles defined as functions with rules for passing messages among roles. LLMs can mediate between roles and between a role and the rules of that role.²⁹ In radiation oncology, automation of workflow with the support of LLMs is advanced,³⁰ but not in palliative care.

A hospice care system, which is the ultimate example of palliative care, suits LLM integration into the workflow.^{31,32} The system would represent the roles of the interdisciplinary team and the roles of patients and their family, as “dignity therapy” illustrates. Dignity therapy helps patients reflect on end-of-life issues. The patient receives nine standard questions which guide a conversation. The dialog is recorded, transcribed, and edited into a legacy document for the patient to use in implementing the last steps of life.³³ However, dignity therapy is underutilized due to a lack of trained staff to help. With an LLM-automated social worker, patients could develop their dignity document.^{34,35}

DISCUSSION

The IEEE literature highlights that AI with Big Data will improve health care. Radiation oncology relies on Big Data, attracts investment, and uses AI extensively. Palliative care relies on human-human interaction, attracts little capital, and uses AI sparingly. Radiation oncology is rich and getting richer. What is the future for palliative care?

Palliative care does not process vast amounts of information in real time, as radiation oncology does. Palliative care relies on natural language communication, and LLMs facilitate natural language communication. As death approaches, the roles of palliative care and natural language

communication increase. LLMs are generalists³⁶ and could support palliative care applications both clinician-facing and patient-facing across every phase of the clinician and patient experience.

The betting AI entrepreneur invests in radiation oncology rather than palliative care. Radiation oncology uses LLMs, and as LLMs are commoditized, their use diffuses into poorer domains, such as palliative care. The radiation oncology patient cannot hope to understand the intricacies of radiation treatment, trusts the doctor, and hopes for a cure. For the hospice patient, by contrast, the doctor is less relevant, and the patient could benefit from LLMs but gets no help in using LLMs. Would the betting person see anything different here? □

REFERENCES

1. R. K. Merton, “The Matthew effect in science: The reward and communication systems of science are considered,” *Science*, vol. 159, no. 3810, pp. 56–63, 1968, doi: 10.1126/science.159.3810.56.
2. P. Zhao, J. Li, and M. Zhang, “Unequal roles of cities in the intercity health-care system,” *Nature Cities*, vol. 2, pp. 198–209, Jan. 2025.
3. R. Rada, *Information Systems and Healthcare Enterprises*. Hershey, PA, USA: IGI Global, 2007.
4. S. C. Lo, M. T. Freedman, J. S. Lin, and S. K. Mun, “Automatic lung nodule detection using profile matching and back-propagation neural network techniques,” *J. Digit. Imag.*, vol. 6, no. 1, pp. 48–54, Feb. 1993, doi: 10.1007/bf03168418.
5. A. J. Thirunavukarasu, D. S. J. Ting, K. Elangovan, L. Gutierrez, T. F. Tan, and D. S. W. Ting, “Large language models in medicine,” *Nat. Med.*, vol. 29, no. 8, pp. 1930–1940, Aug. 2023, doi: 10.1038/s41591-023-02448-8.
6. X. Shi et al., “A bibliographic dataset of health artificial intelligence research,” *Health Data Sci.*, vol. 4, Apr. 2024, Art. no. 0125, doi: 10.34133/hds.0125.
7. H. Berghel, “ChatGPT and AIChat epistemology,” *Computer*, vol. 56, no. 5, pp. 130–137, May 2023, doi: 10.1109/MC.2023.3252379.
8. V. Skaramagkas, A. Pentari, Z. Kefalopoulou, and M. Tsiknakis, “Multi-modal deep learning diagnosis of Parkinson’s disease—A systematic review,” *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 31, pp. 2399–2423, 2023, doi: 10.1109/TNSRE.2023.3277749.
9. A. Kumar and N. P. Singh, “A study on future data science and its impact in health care organizations,” in *Proc. 15th Int. Conf. Comput. Commun. Netw. Technol. (ICCCNT)*, Piscataway, NJ, USA: IEEE Press, 2024, pp. 1–7, doi: 10.1109/ICCCNT61001.2024.10724357.
10. G. B. Anugerah and A. N. Hidayanto, “What factors contribute to the acceptance of artificial intelligence on healthcare sector: A systematic review,” in *Proc. 3rd Int. Conf. Intell. Cybern. Technol. Appl. (ICICyTA)*, Piscataway, NJ, USA: IEEE Press, 2023, pp. 477–482, doi: 10.1109/ICICyTA60173.2023.10428744.
11. A. Sharma and R. Kumar, “Artificial intelligence in health care sector and future scope,” in *Proc. Int. Conf. Innovative Data Commun. Technol. Appl. (ICIDCA)*, Piscataway, NJ, USA: IEEE Press, 2023, pp. 210–214, doi: 10.1109/ICIDCA56705.2023.10100220.
12. A. K. Jha et al., “Implementation of big imaging data pipeline adhering to FAIR principles for federated machine learning in oncology,” *IEEE Trans. Radiat. Plasma Med. Sci.*, vol. 6, no. 2, pp. 207–213, Feb. 2022, doi: 10.1109/TRPMS.2021.3113860.
13. F. J. P. Hoebers et al., “Artificial intelligence research in radiation oncology: A practical guide for the clinician on concepts and methods,” *BJR Open*, vol. 6, no. 1, Jan. 2024, Art. no. tzae039, doi: 10.1093/bjro/tzae039.
14. F. Abuhijla, I. Jaradat, A. Almousa, and R. Abuhijlih, “Impact of big data on radiation oncology practice,” in *Proc. 1st Int. Conf. Cancer Care*

- Inform. (CCI), 2018, p. 59, doi: [10.1109/CANCERCARE.2018.8618145](https://doi.org/10.1109/CANCERCARE.2018.8618145).
15. E. Huynh et al., "Artificial intelligence in radiation oncology," *Nature Rev. Clin. Oncol.*, vol. 17, pp. 771–781, Aug. 2020, doi: [10.1038/s41571-020-0417-8](https://doi.org/10.1038/s41571-020-0417-8).
 16. M. Kawamura et al., "Revolutionizing radiation therapy: The role of AI in clinical practice," *J. Radiat. Res.*, vol. 65, no. 1, pp. 1–9, Jan. 2024, doi: [10.1093/jrr/rad090](https://doi.org/10.1093/jrr/rad090).
 17. S. Connor, "Global atlas of palliative care, 2nd edition," Worldwide Hospice Palliative Care Alliance, London, U.K., 2020, p. 120. [Online]. Available: https://www.iccp-portal.org/system/files/resources/WHPCA_Global_Atlas_FINAL_DIGITAL.pdf
 18. R. T. Sullender and S. A. Selenich, *Financial Considerations of Hospital-Based Palliative Care* (RTI Press Research Report Series). Research Triangle Park, NC, USA: RTI Press, 2016.
 19. M. S. Amroud, P. Raeissi, S. M. Hashemi, N. Reisi, and S. A. Ahmadi, "Investigating the challenges and barriers of palliative care delivery in Iran and the world: A systematic review study," *J. Educ. Health Promotion*, vol. 10, no. 1, p. 246, 2021, doi: [10.4103/jehp.jehp_1325_20](https://doi.org/10.4103/jehp.jehp_1325_20).
 20. C. Barry, P. Paes, S. Noble, and A. Davies, "Challenges to delivering evidence-based palliative medicine," *Clin. Med.*, vol. 23, no. 2, pp. 182–184, Mar. 2023, doi: [10.7861/clinmed.2022-0336](https://doi.org/10.7861/clinmed.2022-0336).
 21. "RePORT: Research portfolio online reporting tools," Nat. Inst. of Health, Bethesda, MD, USA, 2025. [Online]. Available: <https://report.nih.gov/>
 22. M. N. Patel et al., "Machine learning for targeted advance care planning in cancer patients: A quality improvement study," *J. Pain Symptom Manage.*, vol. 68, no. 6, pp. 539–547, e3, Dec. 2024, doi: [10.1016/j.jpainsyman.2024.08.036](https://doi.org/10.1016/j.jpainsyman.2024.08.036).
 23. N. D. Agaronnik, J. Davis, C. R. Manz, J. A. Tulsky, and C. Lindvall, "Large language models to identify advance care planning in patients with advanced cancer," *J. Pain Symptom Manage.*, vol. 69, no. 3, pp. 243–250, e1, Nov. 2024, doi: [10.1016/j.jpainsyman.2024.11.016](https://doi.org/10.1016/j.jpainsyman.2024.11.016).
 24. H. Qiu, A. Li, L. Ma, and Z. Lan, "PsyChat: A client-centric dialogue system for mental health support," in *Proc. 27th Int. Conf. Comput. Supported Cooperative Work Des. (CSCWD)*, Piscataway, NJ, USA: IEEE Press, 2024, pp. 2979–2984, doi: [10.1109/CSCWD61410.2024.10580641](https://doi.org/10.1109/CSCWD61410.2024.10580641).
 25. V. J. Nolan et al., "Incorporating patient values in large language model recommendations for surrogate and proxy decisions," *Crit Care Explorations*, vol. 6, no. 8, p. e1131, Aug. 2024, doi: [10.1097/cce.0000000000001131](https://doi.org/10.1097/cce.0000000000001131).
 26. S. Tripathi, R. Sukumaran, and T. S. Cook, "Efficient healthcare with large language models: Optimizing clinical workflow and enhancing patient care," *J. Amer. Med. Inform. Assoc.*, vol. 31, no. 6, pp. 1436–1440, 2024, doi: [10.1093/jamia/ocad258](https://doi.org/10.1093/jamia/ocad258).
 27. V. L. Yu et al., "Evaluating the performance of a computer-based consultant," *Comput. Programs Biomed.*, vol. 9, no. 1, pp. 95–102, Jan. 1979, doi: [10.1016/0010-468X\(79\)90022-9](https://doi.org/10.1016/0010-468X(79)90022-9).
 28. H. Landi, "HIMSS24: How epic is building out AI, ambient technology for clinicians," *Fierce Healthcare*, Mar. 13, 2024. [Online]. Available: <https://www.fiercehealthcare.com/ai-and-machine-learning/himss24-how-epic-building-out-ai-ambient-technology-clinicians#:~:text=When%20we%20use%20the%20draft,that%20was%20generated%20by%20AI.%22>
 29. K. J. Prabhad, "Integrating large language models for enhanced clinical decision support systems in modern healthcare," *J. Mach. Learn. Healthcare Decis. Support*, vol. 3, no. 1, pp. 18–62, Jun. 2023. [Online]. Available: <https://medlines.uk/index.php/JMLHDS/article/view/23>.
 30. R. Krishnamurthy, N. Mummudi, J. S. Goda, S. Chopra, B. Heijmen, and J. Swamidas, "Using artificial intelligence for optimization of the processes and resource utilization in radiotherapy," *JCO Global Oncol.*, vol. 8, Nov. 2022, Art. no. e2100393, doi: [10.1200/go.21.00393](https://doi.org/10.1200/go.21.00393).
 31. G. Demiris, D. Parker Oliver, and E. Wittenberg-Lyles, "Technologies to support end-of-life care," *Seminars Oncol. Nursing*, vol. 27, no. 3, pp. 211–217, Aug. 2011, doi: [10.1016/j.soncn.2011.04.006](https://doi.org/10.1016/j.soncn.2011.04.006).
 32. A. Suslow, C. Giehl, J. Hergesell, H. C. Vollmar, and I. Otte, "Impact of information and communication software on multiprofessional team collaboration in outpatient palliative care - A qualitative study on providers' perspectives," *BMC Palliative Care*, vol. 22, no. 1, Mar. 2023, Art. no. 19, doi: [10.1186/s12904-023-01141-4](https://doi.org/10.1186/s12904-023-01141-4).
 33. M. Martínez et al., "'Dignity therapy', a promising intervention in palliative care: A comprehensive systematic literature review," *Palliative Medicine*, vol. 31, no. 6, pp. 492–509, 2017, doi: [10.1177/0269216316665562](https://doi.org/10.1177/0269216316665562).
 34. Y. Lim, "Dignity and dignity therapy in end-of-life care," *J. Hospice Palliative Care*, vol. 26, no. 3, pp. 145–148, Sep. 2023, doi: [10.14475/jhpc.2023.26.3.145](https://doi.org/10.14475/jhpc.2023.26.3.145).
 35. E. Wild, J. Weng, D. Pachman, C. Ingram, and M. Felt, "Feasibility and acceptability of virtual dignity therapy for palliative care patients with advanced cancer," *J. Pain Symptom Manage.*, vol. 67, no. 5, pp. e609–e610, 2024, doi: [10.1016/j.jpainsyman.2024.02.032](https://doi.org/10.1016/j.jpainsyman.2024.02.032).
 36. M. Moor et al., "Foundation models for generalist medical artificial intelligence," *Nature*, vol. 616, pp. 259–265, Apr. 2023, doi: [10.1038/s41586-023-05881-4](https://doi.org/10.1038/s41586-023-05881-4).

ROY RADA is a professor emeritus at the Department of Information Systems, ITE Building, University of Maryland, Baltimore County, Baltimore, MD 21250 USA. Contact him at rada@umbc.edu.



Bridging the Gap Between Engineering and Cybersecurity to Better Protect Critical Infrastructures

Joseph Weiss^{ID}, Applied Control Solutions, LLC

James Bret Michael^{ID}, Naval Postgraduate School

This article addresses the need to bridge the gap between engineering and cybersecurity to better protect critical infrastructures such as electric power grids.

In several articles we published in this column, the most recent of which was in January 2024, we noted that although critical infrastructures including electric power grids are vulnerable to unintentional cyber incidents and cyberattacks, critical infrastructures' control systems make them resilient to such incidents and attacks, that is, the grids function despite the degradation, denial, or destruction of their operational technology (OT) networks.¹

We turn our attention in this article to a gap we observe between the engineering organizations responsible for reliability, functionality, and safety of critical infrastructures on the one hand, and the cybersecurity organizations responsible for the network secu-

rity and other aspects of cybersecurity of such systems on the other.

Digital Object Identifier 10.1109/MC.2025.3566335
Date of current version: 27 June 2025

Prior to publication of the 2006 Gartner research paper introducing the term OT, there was either engineering or IT. The term OT created a hybrid between engineering and IT that has not been properly connected. Engineers come from a physics-based discipline, whereas OT comes from a data-centric discipline.

The term OT is fairly well known in the cybersecurity community. However, OT does not appear to be widely

and job requirements for engineering and cybersecurity professionals. Let's look at these from the perspective of postsecondary education and job requirements.

POSTSECONDARY EDUCATION

Cybersecurity is usually taught as a track of study within computer science baccalaureate degree programs, but those programs typically do not

Modern computer networking technologies provide enhanced productivity and efficiency but often at the expense of risk associated with cybersecurity vulnerabilities.

recognized or understood by engineers working on critical infrastructures. This disconnect is unfortunate, as control systems are neither just engineering nor Internet Protocol (IP) networks but a combination of both. Modern computer networking technologies provide enhanced productivity and efficiency but often at the expense of risk associated with cybersecurity vulnerabilities. Yet, both engineering and cybersecurity organizations continue to act as if the other does not exist. The term OT makes this problem worse as its focus is on networks and not on the issues uniquely associated with control systems and control system field devices and the safety of the physical processes.

This gap in mutual understanding has prevented critical plant processes and control system equipment from being cybersecurity and safe. In the report titled "Who's In Charge of OT Security," Weiss discusses OT-specific vulnerabilities and the need for specialized protection strategies to ensure safe, secure, and resilient operations without compromising efficiency.²

We believe that two of the contributing factors to this situation—a knowledge and culture gap—are the structure of postsecondary education

include courses on engineering except as electives, whereas baccalaureate degrees in electrical, mechanical, chemical, nuclear, systems, and other engineering disciplines treat courses in cybersecurity as electives. And from there, the die is cast, with some subset of graduates of these programs having become knowledgeable in both engineering and cybersecurity. The situation is similar for vocational training, with a heavy emphasis on either engineering or cybersecurity.

However, there are efforts to reduce the knowledge gap. There are short courses available for practitioners from the SANS Institute, Idaho National Laboratory, U.S. Cybersecurity and Infrastructure Security Agency (CISA), and other organizations that focus on the fundamentals or advanced topics of OT assurance. Similarly, there are undergraduate and graduate-level certificate and degree programs that treat engineering and cybersecurity in a unified manner, such as the University of Pittsburgh's undergraduate certificate in Cybersecurity in Emerging Engineering Systems.³ For instance, in addition to completing two courses in cybersecurity and one in artificial intelligence, a student in the certificate program could take the course ECE 1773, Power

Generation, Operation, and Control, which covers the topic of power system security, followed by delving deeper into assurance of critical infrastructure by completing a capstone project. As another example, the Naval Postgraduate School offers the interdisciplinary Cyber Systems and Operations program, with one of the tracks leading to a Master of Electrical Engineering Science (with an emphasis on Electrical Engineering).⁴ Five of the core courses in that track focus on cybersecurity and cyberphysical systems: Introduction to Cybersecurity, Introduction to Cyber Systems and Operations, Cyber Network and Physical Infrastructures, Network Security, and Introduction to Cyber Physical Systems. The students in that program have plenty of electives that they can take in engineering related to cyberphysical systems and control and can select a thesis topic centered on some aspect of defending critical infrastructure from physical or cyberattack.

JOB REQUIREMENTS

A recent job solicitation from a medium-size water utility was seeking junior, mid-level, and senior engineers. The solicitation stated the essential functions were to "Assist with or lead providing electrical engineering and technical support to ensure reliable operation of the utility's supervisory control and data acquisition (SCADA)-controlled facilities including remote

DISCLAIMER

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright annotations thereon.

terminal units, programmable logic controllers, programmable automation controllers, associated industrial communications, networking equipment and protective relaying equipment." Even though the engineers were responsible for the industrial networks and networking equipment (these would be considered OT networks), the engineers were not responsible for cybersecurity of those networks as there was no mention of cybersecurity in the job descriptions. Moreover, there was no mention of the term OT nor any consideration with working with the network security organization.

Another recent job solicitation was from a large electric utility for an OT cybersecurity senior analyst. According to the solicitation, the analyst would be part of a team consisting of skilled OT cybersecurity professionals to ensure the cybersecurity resilience and regulatory compliance of the utility's industrial operational sites. The analyst should have a computer science (not engineering) degree. The focus would be on identifying vulnerabilities and assessing risks to uphold and continuously improving the security posture of industrial control systems (ICS) and OT environments. There was no mention of understanding the physical process or working with the engineering organizations.

Let's look at some examples in which engineering organizations designed equipment with cyber vulnerabilities, and network security organizations created reliability and safety issues with control systems and facility equipment.

ENGINEERING WITHOUT NETWORK SECURITY INPUT

In the early 2000s, heat waves caused failures of many electric distribution transformers. Consequently, one of the substation suppliers published the following advertisement:

"Equipped with an Ethernet interface and Webserver,

Vendor A Unit Substations now provide simple, affordable access to power system information—including transformer coil temperatures—using a standard Web browser. The preengineered equipment connects to a customer's existing Ethernet LAN much like adding a PC or printer. Unit substations include a Temperature Controller, which provides remote

issues. For example, one of the data sheets indicates Bluetooth is enabled by default, assuming that distance will mitigate any cyber vulnerabilities—a questionable assumption at best. In the 20 June 2023 issue of ControlGlobal (www.controlglobal.com), you will find the text "Updated Pressure Transmitter Increases Technician Safety and Makes Work Faster and Easier—powerful features like graphical backlit display, Bluetooth connectivity,

Cybersecurity is usually taught as a track of study within computer science baccalaureate degree programs, but those programs typically do not include courses on engineering except as electives.

access to transformer data, in addition to its primary role in controlling cooling fans. With a simple click of a mouse, it is easy to monitor transformer coil temperatures per phase and verify cooling fan status at a glance. These new capabilities make it possible to correlate circuit loading with transformer temperatures to extend equipment life. The Ethernet interface and webserver are embedded in this vendor's medium and low voltage switchgear, unit substations, motor control centers, switchboards, and panels."

Based on reviewing the 2023 instrument data sheets on digital pressure transmitters from four major U.S. and international process sensor manufacturers, all of which were involved in industry cybersecurity activities, none of those data sheets contained the terms cyber, security, password, authentication, or encryption. However, the data sheets from all four manufacturers contained many instances of the term "remote" because their digital pressure transmitters support remote connectivity. Engineering electric power systems with such a capability introduces network security

easier to navigate user interfaces, level and flow specific configurations and diagnostic features allow you to perform commissioning, maintenance and troubleshooting tasks faster than ever."

NETWORK SECURITY WITHOUT ENGINEERING INPUT

The security group was scanning data center assets. Because the effort was successful, the security group decided, with no input from Operations, to extend the scanning to substations having to meet North American Electric Reliability Corporation Critical Infrastructure Protection standards. These were 230/500 kilovolt transmission substations critical to the reliable and safe operation of the regional grid. The security group had no previous experience with scanning substations. No notification was given for the scanning to the internal support groups that were responsible for the substation networks. That is, the OT team was not notified that substation scanning was being done by the IT organization. Following the scans, the relay communications were impacted, but the distributed network protocol (DNP, an OT communication protocol) polling was working properly and the

networks in most substations were stable—SCADA was unaware of the problems. The port scanning by this tool caused the IEC 61850 standard's generic object-oriented substation event real-time protocol operation of the relays to stop and suspend operation at the CPU (two different relay suppliers) but left the DNP/nonreal time operations alone—the worst possible circumstance. To clear the trouble and restore operations, each relay had to be cut out and rebooted. More than four hundred high-voltage relays were affected. All the devices in each substation were affected at the same time in every case. Without knowing that a security scan was initiated, it looked like a distributed denial-of-service attack resulting in equipment malfunctions. Given the appropriate circumstances, this IT-caused problem could have created a major Western States outage.

On 16 March 2022, the National Institute of Standards and Technology (NIST) issued NIST Special Publication 1800-10 "Protecting Information and System Integrity in Industrial Control System Environments: Cybersecurity for the Manufacturing Sector." The NIST report states "in this project, the focus was on the engineering workstations and not on the manufacturing components. It is acknowledged that many of the device cybersecurity capabilities may not be available in modern sensors and actuators."⁵

The lack of cybersecurity capabilities in legacy control systems means that CISA cybersecurity recommendations, such as system hardening, removing default passwords, patching, and penetration testing that are relevant to IP devices can be harmful to process sensors and actuators that do not have the IP stacks and other security capabilities to utilize the CISA recommendations. Antivirus software has shut down older legacy control systems. Addressing Microsoft tools and scripts are irrelevant to process sensors and actuators that do not use

Microsoft operating systems, of which there are many. There have been numerous cases in which inappropriate IP network technologies and testing have shut down or damaged control system devices, which is the concern about CISA's guidelines.

There appear to be many "OT experts" who know little about control system devices yet provide OT/ICS advice. Control system field devices are technically and operationally different from IP networks. Consequently, inappropriate advice from these "OT experts" can, and has, harmed control system field devices and shut down facilities.

Engineering and cybersecurity professionals need to understand each other's strengths and limitations and work together "to make a whole." This needs to start in postsecondary education and be properly recognized in job requirements for people involved in developing and sustaining critical infrastructures including electric power grids. Moreover, continuous education and training in control system cybersecurity is needed to minimize the inappropriate advice being dispensed by "OT cybersecurity experts" who do not understand the control system field devices, but also to keep up with the changes in technology and best practices of both disciplines. Beyond workforce upskilling, senior leaders need to ensure that the engineering and security organizations support, not ignore, each other. □

REFERENCES

- M. T. Swearingen, J. B. Michael, J. Weiss, and R. Radvanovsky, "Resilient without zero trust," *Computer*, vol. 57, no. 1, pp. 120–122, Jan. 2024, doi: [10.1109/MC.2023.3329672](https://doi.org/10.1109/MC.2023.3329672).
- J. Weiss, "Who's in charge of OT security," Sam Houston State Univ., Huntsville, TX, USA, Report No. IHS/CR-2024-1033, 2024. Accessed: Apr. 28, 2024. [Online]. Available: https://ihsonline.org/Portals/0/Tech%20Papers/2024_Papers/Weiss_Whos_in_Charge_of_OT_Security.pdf?ver=dDp8Pyt-VOTTz8UqIu687gg%3d%3d
- "Cybersecurity in emerging engineering systems, undergraduate certificate," Univ. of Pittsburgh, Pittsburgh, PA, USA. Accessed: Apr. 28, 2025. [Online]. Available: https://catalog.upp.pitt.edu/preview_program.php?catoid=225&poid=72696&returnto=23311
- "Cyber systems and operations (CSO) program, in 2024–24 course catalog," Naval Postgraduate School, Monterey, CA, USA. Accessed: Apr. 28, 2025. [Online]. Available: <https://nps.smartcatalogiq.com/en/current/academic-catalog/departments/department-of-information-sciences/cyber-systems-and-operations-curriculum-326/cyber-systems-and-operations-cso/>
- M. Powell et al., "Protecting information and system integrity in industrial control system environments: Cybersecurity for the manufacturing sector," Nat. Inst. of Standards and Technol., Gaithersburg, MD, USA, Special Publication NIST SP 1800-10, Mar. 2022. Accessed Apr. 28, 2025. [Online]. Available: <https://csrc.nist.gov/pubs/sp/1800/10/final>

JOSEPH WEISS is a professional engineer and managing partner with Applied Control Systems, LLC, Cupertino, CA 95014 USA. Contact him at joe.weiss@realtimeacs.com.

JAMES BRET MICHAEL is a professor in the Department of Computer Science and Department of Electrical and Computer Engineering, Naval Postgraduate School, Monterey, CA 93943 USA. Contact him at bmichael@nps.edu.



Cyberphysical Systems in Control: Risks of Digital Transformation



Dimitrios Serpanos^{ID}, Computer Technology Institute and Press “Diophantus” and University of Patras

Panayotis Antoniadis^{ID}, NetHood and Internet Society Switzerland Chapter

Digital transformation outpaces the management of associated risks. As incidents and empirical data indicate significant dangers at many fronts, contingencies need to be addressed to achieve resilience, protect privacy, and increase benefits for all.

Our digital life has been dramatically increasing in the last decades. Powered by the advancing speeds of the Internet, the growth and availability of cloud technologies, mobile systems,

wide deployment of cyberphysical systems, and the Internet-of-Things, the digital transformation (digitization) of our everyday services and applications is shaping our daily activities from banking and driving to health and entertainment. Digitization is also penetrating the operation of infrastructures around the world, from electrical grids and transport networks to industrial floors and defense systems.

The increasing rate of the transformation leads to an increasing portion of user and operator time spent on these services. Although they provide convenience, speed,

and effectiveness, they have side effects that include or lead to risks, personal or social. This is due to their way of deployment, since they are often deployed as substitutions to physical services, rather than alternatives, without considering their acceptance by users or effects in the case of their unavailability. Users often resist digital service adoption for a range of reasons, including lack of skill,

dislike to change habits, and doubts or fear of mismanagement or unexpected developments in information sharing, as in the case of organization mergers and the consolidation of customer/user data.

Considering the interests and priorities of digital service producers and consumers, it becomes increasingly clear that at several fronts, consumers and producers are not in

capabilities and approaches to address this additional burden depending on many parameters. The level of education and digital skills, the lack of infrastructure in parts of cities and in rural areas, and the high cost—for many—of smart devices leads to a digital divide among people. The detection and recognition of the digital divide is not new and constitutes an increasing concern for organizations

credential leakage, exploitation, and crime sometimes, demonstrating clearly not only the fact that the human is the weak link in cybersecurity but that the problem of providing digital services to citizens is a complex landscape that includes digital technologies, sociology, management, and legal issues that need to be addressed altogether to lead to effective, inclusive, safe, and secure digital services.

On the other hand, digital natives, and especially the younger generation, become increasingly addicted to digital interfaces while they are losing cognitive and social skills, such as reading a book, orienting themselves in a city, or having a focused discussion, and recent reports on psychological effects of digital platforms on young people are very worrisome.² Moreover, the effort saved by artificial intelligence (AI) agents and other automation systems is, in many cases, meaningful effort, like the thought put into writing a critical essay, and its replacement is equivalent to having a robot perform physical exercise on one's behalf. Most importantly, the time saved from doing the digital tasks faster is not invested in healthy physical or mental activities, in general, but in further obsessive participation in social groups and use of digital entertainment in social media platforms. It appears that the more efficient digital services become, the less time people have available. At the same time, a culture is growing that one cannot exist without a smartphone in today's society.

Clearly, a balanced approach is necessary in many technological and social directions so that the privileged with increasing access to digital services are motivated for nondigital activities, while the nonprivileged or excluded have alternative, nondigital access options to services to lead a normal life.

sync. Incidents like the recent Spanish blackout¹ that led to inability to conduct even simple daily functions, like shopping and banking, increase the worries of citizens about the robustness of infrastructures and services. The high dependence on digital services creates additional vulnerabilities in public health because of addiction and limited physical activity. Information and awareness are also threatened because of the polarizing effects of social media and the tremendous power of digital platforms to manipulate behavior and perception. Importantly, digitization has a strong environmental impact, for example, causing increased CO₂ emissions, which needs to be analyzed and evaluated further. These risks and concerns make many users wary and resistant to the transformation. Although the benefits of digitization for society are significant in the long run, it is critical to address its potential risks early to increase its acceptance widely.

USER PARTICIPATION AND THE DIGITAL DIVIDE

Digitization shifts significant burden and effort to users; this holds for all services, ranging from e-government services to workers in the field. Importantly, users have different

and governments who take education and awareness measures, like continuous education and reskilling, in order to address it. However, the digital divide is a reality and expresses itself in many directions with differing participation, engagement, and effective use of services.

The lack of inclusiveness is a concern for the case of people with disabilities as well. Although many services are designed with appropriate interfaces for disabled persons, it is often the case that services do not provide support for all, and sometimes for most, of the users in the category of disabilities and special needs; significant advances to human-computer interface technologies are necessary to enable inclusion, security, and safety in the use of digital services. Importantly, some of these services have no alternatives to the service except the digital option, which, effectively, excludes groups of people. Consider, as an example, the case of mandatory electronic-only submission of a tax return of an unskilled or disabled person that cannot use the electronic filing system. The common approach to resolve the issue, since the burden of compliance is with the citizen, is to use a third intermediary party to do the submission. This may lead to

CYBERSECURITY AND SAFETY

Cybersecurity, including availability, and safety are high-priority concerns due to digitization. The increasing

population of digital service users and operators is augmenting their attack surface as in the aforementioned case of unskilled users.

The cybersecurity risks of digitization are fairly well-known and experienced, unfortunately, through various incidents such as malware attacks that lead to data exfiltration and/or ransom payments, denial of service attacks, insider data misuse, and others. The human-in-the-loop is the most vulnerable link in the operational chain and is often the starting target of attackers. The Global Anti-Scam Alliance estimates that US\$1.03 billion is lost per year due to online scams,³ while the total cost of cybercrime has been estimated up to US\$10.5 trillion in 2025.⁴ Governments and organizations face significant financial burden to address these risks and associated incidents. However, the increasing digitization provides attractive targets to cybercriminals continuously, simply because increased data records and transactions lead to increased probability of data leaking. The ability of attackers to hack humans leads to attackers exploiting legitimate users and operators. The problem is even worse when taking into account the complex supply chains for developing systems and services, where contractors of different levels and capabilities are involved; the problem is becoming increasingly acute, considering the fast turnaround time requirements and the worldwide lack of skill, especially in cybersecurity.

Robustness to service provisioning is necessary for operational continuity at organization, government, and social levels; this requires the anticipation of operational scenarios where digital services are minimal or absent due to incidents. Such robustness requirements have become apparent in cases like the Facebook outage incident where employees could not access offices, buildings, or digital services due to a bug that was triggered during routine network maintenance.⁵ Clearly,

the ability to enter one's own house must be continuously available and not dependent only on digital services, even on the presumably always available cloud.

Who would argue that stairs are not necessary in a skyscraper because elevators are faster and more comfortable? Still, there are many proponents of digitization who ignore the necessity of maintaining different

the exploitation of personal data, and the view "I do not have anything to hide" is clearly a fallacy.

The problem of profiling is becoming increasingly important as service providers restructure and data provided at different sites are combined through repurposing of repositories in organizations or governments. As more complete profiles are constructed by interested providers, the complex

The more efficient digital services become, the less time people have available.

alternatives for basic human activities in the name of technological progress and efficiency.

PRIVACY

The growth of digital services and applications spans all domains from banking to health, from transport to government services. Services and applications store and process transactions of individuals and organizations, including sensitive data. The increasing amounts of related stored data have extended the risks to privacy significantly.

Digital marketing has been leading the way to collecting user information and providing effective personalized services; this is achieved by creating profiles for users. Personalization is considered a value-added service, in general, but at the same time, it is an open question whether it is valuable in many domains. Actually, it raises some sociological questions about the possibility that personalization may be leading to limited choices if/when applied in the wrong domain or with the wrong parameters. In practice, personalization methods can influence humans and lead them to controlled subsets of decisions. Thus, the collection of personal information may be exploited in ways unknown to the subject who may have even agreed to provide their personal information. It is apparent that the risks are high for

supply chain of service development and provisioning creates an additional risk in this context as well. Subcontractors can obtain access to profile data through their participation in the related supply chain, leading to a high-probability risk for personal data leakage and privacy violations. Legal frameworks, like General Data Protection Regulation (GDPR) in Europe, provide defense, but it is very difficult to ensure that systems and processes are GDPR compliant, at least before damage is caused.

Notably, the issue of citizens' and organizations' privacy is linked to national security and (digital) sovereignty. This is becoming a significant and high-priority issue across the world as demonstrated by the TikTok ban in the United States and the EuroStack initiative in Europe, where the significant majority (more than 80%) of digital infrastructure is foreign, leading to European dependence and vulnerabilities.⁶

AI AND MACHINE LEARNING

AI is causing a revolution in digitization. All related technologies, from machine learning (ML) to generative AI, have a strong impact on the development of efficient, effective, and low-cost services ranging from helpdesk and custom support to predictive maintenance in cyberphysical systems. The benefits are significant,

but the quick pace of introducing AI/ML in production systems is much faster than the pace of identifying and managing the risks that are associated with the current employment of AI/ML in digital services.

The AI breakthroughs have disrupted a wide range of technologies and services and initiated a strong discussion on its adoption risks at several fronts: ethical, technical, privacy, societal, economic, and regulatory. Many of these risks have already been manifested in systems where bias is clearly present or safety is compromised. The first efforts to counter these risks are simple patches that are easily circumvented either by alternate systems that one can easily find on the Web or through appropriate manipulation of the prompts. Efforts for explainable and trustworthy AI are in their beginning, and significant effort is still required to attain the necessary levels of ethics, fairness, security, and safety.

The quick adoption of AI in several domains where failures can be catastrophic, as in the case of autonomous vehicles and health systems, raises significant ethical and legal questions, including liability, which require novel legal and regulatory frameworks for emerging systems. Importantly, the operation of AI systems will be significantly impacted by legal and regulatory constraints, indicating the urgent need to address AI risks in a systematic and holistic way, from technology to law, from intellectual property protection of training data to safety.

The accelerating digital transformation worldwide provides significant advantages and cost savings at all domains, from e-government and health to transportation and the industrial floor. However, the

transformation has risks that span areas from social and economic to technical and legal, considering the effects on workforce, innovation, privacy, and others. As the transformation outpaces the management of the associated risks by societies, it is necessary to address risks with several approaches. Proactively, societies should promote cultures for social robustness where digital services are only one of the tools, and for a critical, informed attitude toward technological innovations. Reactively, societies should address technology addiction and dependence with appropriate measures; for example, digital services can be employed only where needed and not where they can be functional in general. The materializing risks of digitization have a significant impact on societies, organizations, and governments, with such results as digital addiction, which affects our mental and physical health directly, especially the younger generations. ■

REFERENCES

1. A. Brezar and C. Skopeliti, "Spain, Portugal and parts of France hit by massive power outage," *Euronews*, Apr. 28, 2025. Accessed: May 4, 2025. [Online]. Available: <https://www.euronews.com/my-europe/2025/04/28/spain-portugal-and-parts-of-france-hit-by-massive-power-outage>
2. "Social media and youth mental health: The U.S. Surgeon General's Advisory," U.S. Public Health Service, Washington, DC, USA, 2023. Accessed: May 4, 2025. [Online]. Available: <https://www.hhs.gov/sites/default/files/sg-youth-mental-health-social-media-advisory.pdf>
3. "Global Anti-Scam Alliance: Protecting consumers worldwide from scams," Global Anti-Scam Alliance (GASA). Accessed: May 1, 2025. [Online]. Available: <https://www.gasa.org>
4. S. Morgan, "Cybercrime to cost the world \$10.5 trillion annually by 2025," *Cybercrime Magazine*, Nov. 13, 2020. Accessed: May 1, 2025. [Online]. Available: <https://cybersecurityventures.com/cyberwarfare-report-intrusion/>
5. S. Moss, "Facebook blames major outage on maintenance work "effectively disconnecting Facebook data centers globally," *Data Center Dynamics*, Oct. 6, 2021. Accessed: May 1, 2025. [Online]. Available: <https://www.datacenterdynamics.com/en/news/facebook-blames-major-outage-on-maintenance-work-effectively-disconnecting-facebook-data-centers-globally/>
6. F. Bria, P. Timmers, F. Gernone, A. Renda, C. Fischer, and O. Grabova, "EuroStack – A European alternative for digital sovereignty," *ceps.eu*, Feb. 13, 2025. Accessed: May 4, 2025. [Online]. Available: <https://www.ceps.eu/ceps-publications/eurostack-a-european-alternative-for-digital-sovereignty/>

DIMITRIOS SERPANOS is the president of the Computer Technology Institute and Press "Diophantus" and a professor at the University of Patras, 26504 Patras, Greece. Contact him at serpanos@computer.org.

PANAYOTIS ANTONIADIS is the co-founder of NetHood Zurich and principal investigator at the Internet Society Switzerland Chapter for the Horizon Europe Next Generation Internet project "NGIO Commons Fund," CH-8005 Zurich, Switzerland. Contact him at panayotis@nethood.org.

Career Accelerating Opportunities

Explore new options—upload your resume today

careers.computer.org



Changes in the marketplace shift demands for vital skills and talent. The **IEEE Computer Society Career Center** is a valuable resource tool to keep job seekers up to date on the dynamic career opportunities offered by employers.

Take advantage of these special resources for job seekers:



JOB ALERTS



TEMPLATES



WEBINARS



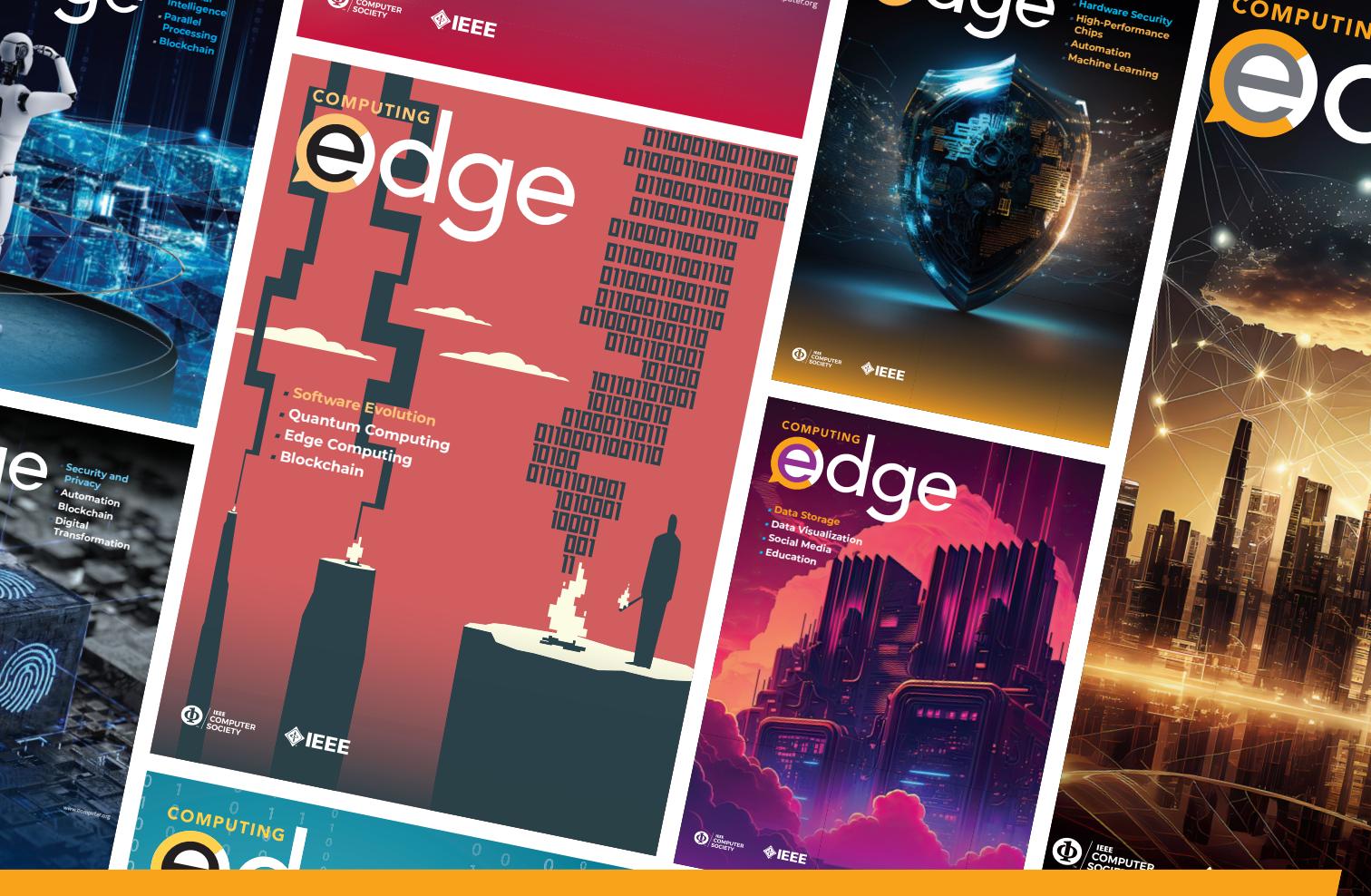
CAREER
ADVICE



RESUMES VIEWED
BY TOP EMPLOYERS

No matter what your career level, the IEEE Computer Society Career Center keeps you connected to workplace trends and exciting career prospects.





ComputingEdge

Your one-stop resource for industry hot topics, technical overviews, and in-depth articles.

Cutting-edge articles from the IEEE Computer Society's portfolio of 12 magazines.

Unique original content by computing thought leaders, innovators, and experts.

Keeps you up to date on what you need to know across the technology spectrum.



Subscribe for free
www.computer.org/computingedge

