COMPUTING edge

- Quantum Computing
- Software Development
- Scientific Computing
- History

AUGUST 2024

www.computer.org

IEEE COMPUTER SOCIETY

IEEE

Get Published in the New *IEEE Transactions on Privacy*

This fully open access journal is now soliciting papers for review.

IEEE Transactions on Privacy serves as a rapid publication forum for groundbreaking articles in the realm of privacy and data protection. Be one of the first to submit a paper and benefit from publishing with the IEEE Computer Society! With over 5 million unique monthly visitors to the IEEE Xplore® and Computer Society digital libraries, your research can benefit from broad distribution to readers in your field.

Submit a Paper Today!

Visit computer.org/tp to learn more.

IEEE COMPUTER SOCIETY | IEEE

COMPUTING edge

IEEE COMPUTER SOCIETY

IEEE

IEEE COMPUTER SOCIETY computer.org

Printed with inks containing soy and/or vegetable oils

STAFF

Circulation: *ComputingEdge* (ISSN 2469-7087) is published monthly by the IEEE Computer Society. IEEE Headquarters, Three Park Avenue, 17th Floor, New York, NY 10016-5997; IEEE Computer Society Publications Office, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720; voice +1 714 821 8380; fax +1 714 821 4010; IEEE Computer Society Headquarters, 2001 L Street NW, Suite 700, Washington, DC 20036.

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

IEEE Computer Society Magazine Editors in Chief

AUGUST 2024 · VOLUME 10 · NUMBER 8

COMPUTING edge

Quantum Computing

Software Development

Scientific Computing

History

Departments

Magazine Roundup

The IEEE Computer Society's lineup of 12 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.

Computer

Venture Building in the Digital Era: Unraveling the Past, Present, and Future of Corporate Innovation

This May 2024 *Computer* article examines the evolution of Venture Building and its symbiotic relationship with digital disruption. The author highlights successful models, offering insights into corporate strategies for navigating digital transformation and fostering entrepreneurial cultures, while anticipating future trends and stressing the importance of further research.

Computing in Science & Engineering

Tensorlab⁺: A Case Study on Reproducibility in Tensor Research

Tensor methods emerge as an important class of basic techniques, generalizing matrix methods to multiway data and models. The authors of this September/October 2023 *Computing in Science & Engineering* article have recently released Tensorlab⁺, which is a downloadable archive of code and data that allows peers to reproduce the experiments reported in their publications on tensor decompositions and applications. They briefly discuss the basic tensor tools and introduce the contents of Tensorlab⁺. The authors elaborate on the steps that were taken to ensure the reproducibility of the experiments and the quality of the code.

IEEE Annals of the History of Computing

Lady Lovelace's Objection: The Turing–Hartree Disputes Over the Meaning of Digital Computers, 1946–1951

Can machines think? Or can they do "whatever we know how to order" them to perform? Should machines be liberated from slavery and given "fair play" to "compete with men in all purely intellectual fields"? Or should this be associated with a fashion that decries "human reason" and a path that "leads straight to Nazism"? In the postwar years, these questions were debated by Alan Turing and Douglas Hartree, who differed in their interpretations of the digital computer as a new piece of science and technology. This January–March 2024 *IEEE Annals of the History of Computing* article examines the Turing–Hartree disputes and draws a parallel between their positions and their perspectives on postwar Britain.

IEEE Computer Graphics and Applications

Integrated Augmented and Virtual Reality Technologies for Realistic Fire Drill Training

The authors of this March/April 2024 *IEEE Computer Graphics and Applications* article propose a novel fire drill training system designed specifically to integrate augmented reality (AR) and virtual reality (VR) technologies into a single head-mounted display device to provide realistic as well as safe and diverse experiences. Applying hybrid AR/VR technologies in fire drill training may be beneficial because they can overcome limitations such as space–time constraints, risk factors, training costs, and difficulties in real environments. The proposed system can improve training effectiveness by transforming arbitrary real spaces into real-time, realistic virtual fire situations, and by interacting with tangible training props.

Published by the IEEE Computer Society

IEEE Intelligent Systems

From Electroencephalogram Data to Brain Networks: Graph-Learning-Based Brain Disease Diagnosis

Many studies exploit brains from the perspective of graph learning to diagnose the nerve diseases of brains. However, many of these algorithms are unable to automatically construct brain function topology based on electroencephalogram (EEG) and fail to capture the global features of multichannel EEG signals for whole-graph embedding. To address these challenging issues, the authors of this March/April 2024 *IEEE Intelligent Systems* article propose an attention-based whole-graph learning model for the diagnosis of brain diseases, namely, MAINS, which can adaptively construct brain functional topology from EEG signals and effectively embed multiple node features and the global structural features of brain networks into the whole-graph representations.

IEEE Internet Computing

Open Experimental Measurements of Sub-6GHz Reconfigurable Intelligent Surfaces

In this March/April 2024 *IEEE Internet Computing* article, the authors present two datasets that they make publicly available for research. The data is collected in a testbed comprised of a custom-made reconfigurable intelligent surface (RIS) prototype and two regular orthogonal frequency-division multiplexing (OFDM) transceivers within an anechoic chamber. The authors discuss the details of the testbed and equipment used, including insights about the design and implementation of their RIS prototype. They further present the methodology they employ to gather measurement samples, which consists of letting the RIS electronically steer the signal reflections from an OFDM transmitter toward a specific location.

IEEE micro

Enabling Artificial Intelligence Supercomputers With Domain-Specific Networks

This article, featured in the March/April 2024 issue of *IEEE Micro*, argues how domain-specific networks are a critical enabling technology necessary for AI supercomputers. In particular, the authors advocate for flexible, low-latency interconnects capable of delivering high throughput across massive scales with tens of thousands of endpoints. Additionally, they stress the importance of reliability and resilience in handling long-duration training workloads and the demanding inference needs of domain-specific workloads.

IEEE MultiMedia

Exploiting Illumination Knowledge in the Real World for Low-Light Image Enhancement

To bridge the gap and benchmark outdoor LLIE tasks, the authors of this January–March 2024 *IEEE MultiMedia* article propose the first outdoor, real-world, 2-D, low-light dataset, dubbed RE2L. Based on RE2L, they propose a semisupervised LLIE framework to further exploit the illumination knowledge from both the signal fidelity constraint and characteristics of normal-light natural images. Experimental results demonstrate that using RE2L for training deep LLIE schemes can improve the model effectiveness, both quantitatively and visually, especially on semisupervised LLIE.

IEEE pervasive COMPUTING
MOBILE SYSTEMS | UBIQUITOUS COMPUTING | INTERNET OF THINGS

Unifying Threats Against Information Integrity in Participatory Crowd Sensing

This October–December 2023 *IEEE Pervasive Computing* article proposes a unified threat landscape

for participatory crowd sensing (P-CS) systems. Specifically, it focuses on attacks from organized malicious actors that may use the knowledge of P-CS platform's operations and exploit algorithmic weaknesses in AI-based methods of event trust, user reputation, decision-making, or recommendation models deployed to preserve information integrity in P-CS. The authors emphasize on intent driven malicious behaviors by advanced adversaries and how attacks are crafted to achieve those attack impacts.

IEEE SECURITY&PRIVACY

Unleashing Malware Analysis and Understanding with Generative AI

Dissecting low-level malware behaviors into human-readable reports, such as cyber threat intelligence, is time-consuming and requires expertise in systems and cybersecurity. This article, featured in the May/June 2024 issue of *IEEE Security & Privacy*, combines dynamic analysis and artificial intelligence-generative transformation for malware report generation, providing detailed technical insights and articulating malware intentions.

IEEE Software

Innovating Industry with Research: eknows and Sysparency

The authors of this article in the May/June 2024 issue of *IEEE Software* present the multi-language software platform *eknows* for building reverse engineering tools and documentation generators as a concrete example of how to successfully translate research on software analysis into innovative products and services. Platform development includes domain-specific requirements and an architecture supporting reuse of components.

IT Professional

Conceptual Framework for Software Change

Ever since the invention of software, change has been a destabilizing factor. Although many new software changes are being applied, the terminologies used to describe them are often inconsistent. This restricts practitioners to designing and evaluating their changes. This March/April 2024 *IT Professional* article aims to develop a conceptual framework of software change based on six main dimensions regarding the source, essence, and consequences of software change. To evaluate the proposed framework, benchmarking is applied against selected 11 previous studies.

Editor's Note

Learning from the Past to Accelerate the Future

Human advancement can be summarized as a series of failures and successes. People test new approaches, fail, and learn from their failures. Quantum computing is a prime example of combining failures and learning to create something new and extraordinary. This issue of *ComputingEdge* explores how to leverage quantum computing to accelerate systems. The articles emphasize the importance of learning from past approaches to enhance future efforts, from applying lessons from ancient empires to using AI in software development. The issue also discusses the importance of improving scientific computing accessibility.

To realize the potential of quantum computing, engineers must determine how to integrate it with existing systems. In "Monetizing Quantum Computing," from *IT Professional*, the author maps the quantum computing landscape, exploring potential growth, investment, and value generation. The *Computing in Science & Engineering* article, "Quantum Computing and High-Performance Computing: Compilation Stack Similarities," illustrates how quantum computing can accelerate HPC applications using a software stack for integration.

AI and quantum computing can advance HPC and software development—with the right approach. The authors of "Accelerating HPC With Quantum Computing: It Is a Software Challenge Too," from *Computing in Science & Engineering*, describe their vision for an integrated ecosystem that combines HPC and quantum software stacks into one system, simplifying user experience. The article, "Low Code for Smart Software Development," from *IEEE Software*, outlines the promise and perils of using AI enhanced low-code environments to quickly deliver software solutions.

To make scientific computing more accessible, engineers must break down barriers between researchers and intricate science software. In the *IT Professional* article, "On the Role of Computer Languages in Scientific Computing," the authors dissect the complexity and roles of computer languages in scientific computing and offer guidelines for choosing the right language. *Computer*'s article, "Democratizing Science Through Advanced Cyberinfrastructure," identifies barriers to accessing cyberinfrastructure in scientific research and how to address them.

Learning from historical mistakes, successes, and discoveries is an important step in future advancement. The authors of *Computer* article, "Byzantine Empire and Its Generals: An Ancient Empire Back to Life in Computer Security" revisit lessons from the Byzantine Empire's reign and apply them to today's problem of software systems security. In "Mathematics, Logic, and Engineering in Computing," from *IEEE Annals of the History of Computing*, the authors trace the origins of computing back to early approaches to logic, mathematics, and engineering in ancient civilizations.

EDITOR: Nir Kshetri, nbkshetr@uncg.edu

DEPARTMENT: IT ECONOMICS

This article originally appeared in IT Professional vol. 26, no. 1, 2024

Monetizing Quantum Computing

Nir Kshetri, *University of North Carolina at Greensboro*

Quantum computers demonstrate significant speed advantages over classical computers in specific tasks. This article explores the current quantum computing landscape, encompassing investment, market growth, and the potential value generation across key industries.

The next frontier in computing, quantum computers, exhibit remarkable speed advantages in performing specific tasks when compared to classical computers. In October 2023, China's national newspaper *Global Times* reported that Jiuzhang 3.0 quantum computer developed by University of Science and Technology of China can solve an extremely difficult mathematical problem known as Gaussian boson sampling 10 quadrillion (10^{15}) times faster than the world's fastest supercomputer "Frontier" (https://www.globaltimes.cn/page/202310/1299679.shtml).

Increasing the computational capacity of a classical computer necessitates roughly doubling the number of transistors addressing a problem. In contrast, a quantum computer's computational capacity has the potential to increase twofold with the addition of a single quantum bit (qubit), the fundamental unit of quantum information, akin to the classic binary bit. (https://tinyurl.com/yc3uvdze). It is obvious that quantum computers' remarkable speed in handling specific tasks is likely to create significant economic and social value.[1] Quantum technologies are thus viewed as a potential solution to address global challenges such as climate change and accelerated drug delivery.[2]

The technology's value creation potential is significantly heightened when integrated with other technological advancements like machine learning and edge computing. Machine learning typically involves numerous parameters and extensive training data, but quantum machine learning can achieve comparable accuracy with fewer parameters. In a preliminary study conducted with Hyundai, researchers developed quantum machine learning algorithms capable of distinguishing between ten road signs in controlled laboratory experiments. Their quantum model employed a mere 60 parameters to achieve the same accuracy as a classical neural network using 59,000 parameters. The quantum algorithms also demand significantly fewer training iterations.[3] Likewise, a paradigm shift in how data is managed and utilized can happen if quantum computing and edge computing are combined. The colossal volume of raw data gathered by sensors is beyond the capabilities of today's classical computing for real-time analysis, but quantum computing can make it feasible (https://www.ibm.com/downloads/cas/KOQZNQPL).

In this article, we discuss the current quantum computing landscape including investment and market growth in this industry. We also delve into quantum computing's capacity to generate value across key industries.

THE QUANTUM COMPUTING LANDSCAPE: INVESTMENT AND MARKET GROWTH

Current State and Ongoing Developments

Current quantum computing technologies are described to be in the noisy intermediate-scale quantum (NISQ) era, which are characterized by quantum devices with few useful qubits and high error rates (https://thequantuminsider.com/2023/03/13/what-is-nisq-quantum-computing/). Practical computations are only achievable on quantum machines with a significant number of qubits, a technology not yet accessible due to the complex hardware development. Data storage in quantum computers involves qubits, created using diverse technologies such as superconducting rings, optical traps, and light photons. Cooling requirements range from near absolute zero to room temperature.[3]

1520-9202 © 2024 IEEE
Digital Object Identifier 10.1109/MITP.2024.3356111
Date of current version 4 March 2024.

Researchers have provided mathematical evidence demonstrating the considerable advantages of quantum computers over current classical counterparts. These advantages are most pronounced in simulating quantum physics and chemistry and in breaking public-key cryptosystems crucial for securing sensitive communications, including online financial transactions. Other contemplated use cases are generally either marginal, speculative, or encompass elements of both[3] and remain in the realm of experimentation and hypothesis.[4]

Despite restricted qubit quantities and elevated error rates, NISQ computers have the capability to execute valuable computations (https://tinyurl.com/3s7e2uwe). There is a focus among developers and users on the realistic possibility of achieving a near-term quantum advantage using current NISQ.[5] In the current prebreakthrough era, quantum applications often employ hybrid algorithms, blending classical and quantum computing capabilities, with classical processors handling part of the computational workflow. An alternative approach includes classical algorithms simulating a quantum system's behavior on a classical processor, referred to as quantum-inspired algorithms.[6]

In certain use cases, quantum models are already exhibiting an edge over purely classical methods, albeit not a significant one.[6] Weather forecasts, relying on simulations using current data, are known for their errors. Achieving more accuracy requires assessing numerous parameters and their interactions, a task exceeding standard computers' capacity. Quantum computers, with their ability to handle multiple parameters, hold promise for transformation. BASF, a German chemical company, is incorporating Paris-based quantum computing startup PASQAL's technology into weather modeling to seek quantum advantages over classical methods.[6]

Major players in the technology sector continue to advance their quantum capabilities. For instance, technology giants such as Alibaba, Amazon, Google, IBM, and Microsoft have already introduced commercial quantum computing cloud services to the market, known as quantum computing as a service (QCaaS).[4] As early as in 2016, IBM provided access to a 5-qubit quantum computer for researchers through the cloud (https://www.protocol.com/manuals/quantum-computing/noisy-intermediate-scale-nisq). Some are providing industry-specific solutions. Microsoft's Azure Quantum Elements, unveiled in June 2023, is a new computing service aimed at facilitating the R&D of novel materials by chemical companies. Leveraging a blend of existing quantum computers, artificial intelligence, and conventional high-performance computing systems, this service allows chemical firms to simulate extensive permutations of atom combinations. It entails utilizing computers to virtually explore potential new materials and subsequently predict how these materials would interact with the physical world (https://www.reuters.com/technology/microsoft-says-new-computing-service-chemicals-can-slash-rd-time-2023-06-21/).

The Quantum Computing Market

The market size of quantum computing is currently small but experiencing rapid growth. According to the International Data Corporation (IDC), the global quantum computing market size was $1.1 billion in 2022 (https://tinyurl.com/4yuee9ws). IDC expects the global quantum computing market to reach $7.6 billion by 2027[5] and $50.22 billion by 2035 (https://tinyurl.com/4yuee9ws).

Currently, the cost of a quantum computer is quite high, ranging from several million to tens of millions of dollars (https://ts2.space/en/how-much-does-1-quantum-computer-cost/). Thus, in the near term, only a few firms will develop or possess quantum computers. Instead, as noted previously, a cloud-computing-style model is emerging, where companies lease access to quantum machines from specialized providers, akin to the way businesses currently procure computing resources from AWS, Google Cloud, and Microsoft Azure.[7] QCaaS allows businesses to harness quantum computing capabilities within budget constraints (https://techwireasia.com/2023/03/equinix-to-offer-quantum-computing-as-a-service/). As noted previously, major technology companies have introduced QCaaS.

While actual commercial use of quantum computing has not yet started, research and analyst firm Enterprise Strategy Group's 2023 Technology Spending Intentions Survey found that a pilot or testing phase to assess practical usefulness of quantum computing was underway for about 28% of enterprises (Figure 1). The survey also found that more than half of the firms had not yet piloted or tested but had researched or shown interest in this technology.

Investment in the Quantum Computing Industry

The investment in the quantum computing industry is witnessing a rapid increase. As reported by data analytics and consulting company GlobalData, venture capital funding for quantum computing startup reached $1.62 billion in 2022 (https://www.investmentmonitor.ai/news/commercial-scale-quantum-computing-unlikely-before-2027/?cf-view). The collective annual investment

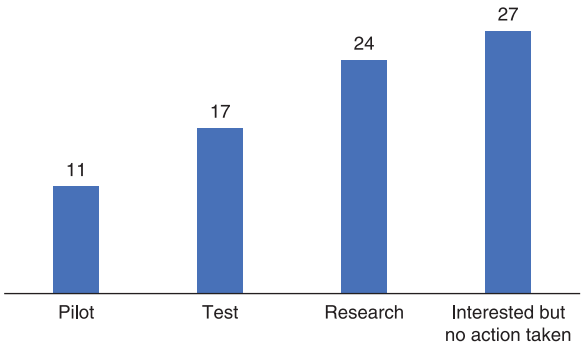


FIGURE 1. Quantum computing readiness of enterprises (percent at different stages). Data source: Enterprise Strategy Group's 2023 Technology Spending Intentions Survey (https://tinyurl.com/mprfxb).

in quantum startups that year reached a record high of $2.35 billion (https://tinyurl.com/aem5ctc5). IDC projects that investments in the quantum computing market will reach approximately $16.4 billion by 2027.[5]

It is worth highlighting the increasing enthusiasm for quantum computing among government agencies worldwide (Figure 2). As of August 2023, 14 entities (comprising 13 countries and the European Union) had unveiled long-term quantum initiatives expected to fund quantum computing research with billions of dollars.[5]

Such funds are also being used to facilitate collaborative research in quantum computing. For instance, in April 2023, the Japanese government announced a plan to invest $31.7 million in a cloud-based quantum computing expansion project led by the University of Tokyo. Over the next five years, the Ministry of Economy, Trade, and Industry would provide funding for this initiative. As of that time, the University used a 27-qubit IBM quantum computer. The government allocated funds to enhance accessibility to the 127-qubit IBM model in the cloud at the University of Tokyo.[8] The project is anticipated to facilitate the utilization of quantum computing by corporations and academic institutions, fostering research collaborations (https://www.u-tokyo.ac.jp/focus/en/features/z1304_00193.html).

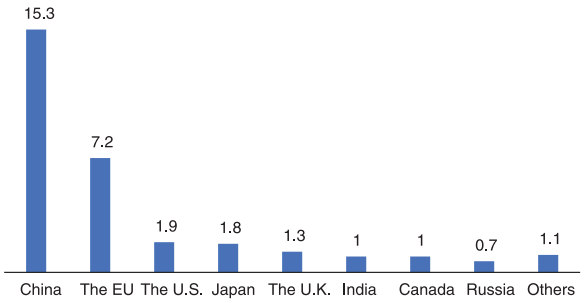


FIGURE 2. Public funds committed for quantum computing investments (as of September 2022, US$, billion). Data source: McKinsey.[2]

QUANTUM COMPUTING'S VALUE CREATION POTENTIAL ACROSS KEY INDUSTRIES

In this section, we explore potential applications within select industries that research indicates could experience significant short-term advantages from quantum technology: pharmaceuticals, chemicals, automotive, and finance. When considered collectively and conservatively, these sectors could potentially unlock a value ranging from $300 billion to $700 billion.[4] Key insights and predictions regarding the value-generating potential of quantum computing in these industries are presented in Table 1.

Pharmaceuticals and Health Care

The quantum computing in health-care market is estimated to reach $1 billion by 2030.[9] With the potential to cut $35 billion in annual R&D expenses for drug discovery and increase $920 billion in annual branded pharmaceutical revenues, quantum computing is anticipated to generate between $35 billion and $75 billion in annual operating income for end users.[10]

Prominent market players are adopting strategic partnerships to address evolving end-user requirements in this sector. In April 2023, IBM joined forces with Moderna, Inc. to utilize quantum computing in mRNA research, aiding the exploration of novel therapeutics.[9] Likewise, Google, in collaboration with Bayer AG, is harnessing quantum computing for early-stage drug discovery, fostering innovation in pharmaceutical development.[9]

Quantum computing has the potential to transform many domains of health care and pharmaceutical industries. In drug design, the process revolves around pinpointing the precise drug target, such as a protein, DNA, or RNA associated with a specific disease, and creating a molecule for safe and effective modification. Despite the abundance of potential targets and molecules, the current approach heavily relies on trial and error, resulting in a time-consuming and expensive venture. Qubit Pharmaceuticals, a Paris-based startup, employs hybrid quantum algorithms for digital twinning of drug molecules, allowing precise simulations of interactions and behavior prediction. This approach

TABLE 1. Quantum computing's effect on some key industries.

Industry	Some key areas likely to be impacted	Market size/sample economic impact
Pharmaceuticals and health care	R&D, early-stage drug discovery	Quantum computing in health-care market: $1 billion by 2030 Drug discovery: Potential to cut $35 billion in annual R&D expenses Expected increase in annual branded pharmaceutical revenues: $920 billion Expected increase in annual operating income $35-75 billion
Chemical	R&D, production, supply chain management, catalyst design	Estimated annual gain: $20–$40 billion
Finance	Portfolio, risk management, attracting and retaining customers	Potential for reducing capital reserves by 15% Expected increase in operating income: $40–70 billion.
Automotive	R&D, product design, supply-chain management, production, optimization of mobility, and traffic control.	Quantum computing market size: $143 million in 2026; $5.2 billion by 2035 Potential annual value generation: $10-$25 billion

eliminates the need for synthesis, potentially reducing drug screening time by half and cutting required investments by 10-fold.[6]

Quantum computing also has a potential to bring a dramatic improvement in conventional cancer treatments, such as radiotherapy and chemotherapy. For instance, radiotherapy employs radiation to eliminate cancerous cells or inhibit their growth. It's essential to create a radiation plan that minimizes damage to healthy tissues and organs, which presents intricate optimization challenges with multiple variables. Quantum computing can aid in simulating numerous scenarios in each iteration, allowing health-care experts to run multiple simulations simultaneously and develop an optimal radiation approach.[9]

Chemical

Quantum advantage holds significant implications, particularly in the context of chemical R&D. It can also enhance production and optimize supply chains. With an annual production budget of $800 billion, the chemicals industry stands to gain between $20 billion and $40 billion in value by achieving a 5%–10% increase in efficiency.[4] For instance, if quantum simulation allows researchers to accurately model material interactions on a larger scale without relying on the imprecise heuristic methods currently in use, companies could potentially cut down or entirely eliminate costly and time-consuming laboratory procedures, like in situ testing. Some companies, such as Zapata Computing, are already placing their bets on the notion that quantum-advantaged molecular simulation will not only lead to substantial cost reductions but also facilitate the development of superior products that can reach the market more quickly.[10]

Half of the production in this industry, valued at $400 billion, relies on catalysts.[4] Quantum computing can be used in production to improve catalyst designs. The technology can be used to create new catalysts and improve existing catalysts. These catalysts have the potential to achieve energy savings in current production processes. As an example, a single catalyst can result in efficiency gains of up to 15%, and pioneering catalysts might facilitate the transition from petrochemicals to more sustainable feedstock or the conversion of carbon into usable CO_2.[4]

Finance

The finance industry has consistently relied on computing speed as a means of gaining an advantage, with hedge funds particularly focused on securing millisecond advantages in price information acquisition.[11] Quantum computing is thus likely to lead to a significant increase in data processing speed. For instance, TerraQuantum, a Swiss startup, works with fintech firm Cirdan Capital to apply quantum-inspired algorithms for pricing exotic options, achieving a 75% increase in pricing speed over traditional methods.[6]

Unsurprisingly, global financial giants, including Allianz, Barclays, Citigroup, Goldman Sachs, JPMorgan,

and Mizuho, as well as a number of national and regional companies are actively researching quantum computing (https://tinyurl.com/mrzvrjua). Turning quantum computing investments and endeavors into financial returns is the goal for these companies.

The creation of analytical models that can efficiently and accurately sort through extensive behavioral data to determine the products essential for specific customers in near real-time is a complex endeavor. This constraint prevents financial institutions from delivering preemptive product recommendations with optimal feature selection in an agile manner (https://tinyurl.com/yc3uvdze). Likewise, due to the high inaccuracy of fraud detection systems, financial institutions tend to adopt an overly risk-averse approach. The customer onboarding process, lasting 12 weeks, is facing growing resistance from consumers. Financial institutions that fail to engage with new customers swiftly are losing them to more agile competitors (https://tinyurl.com/yc3uvdze). The use of quantum computing is likely to address and overcome these challenges. The technology has the potential to revolutionize customer targeting and prediction modeling by surpassing current limitations related to complex data structures. Its data modeling capabilities are anticipated to excel in pattern recognition, classification, and prediction tasks (https://tinyurl.com/yc3uvdze).

Quantum computing is also anticipated to have an impact on portfolio and risk management. An instance of this is the successful quantum optimization of loan portfolios, which could empower lenders to improve their services, potentially decreasing interest rates and boosting capital availability.[4] Quantum computing technology is maturing, leading to improved model accuracy and increased resilience against extreme tail events such as those that might arise once in every 50 years.[10] Multiverse Computing, a Spanish quantum startup, has partnered with Spanish multinational financial services company Banco Bilbao Vizcaya Argentaria, S.A. to improve investment portfolio optimization, addressing the common challenge of accounting for external factors' impact on asset performance in finance. The experiment showcased that Multiverse's quantum-inspired methods sped up calculations, maximizing profitability while minimizing risk.[6] The potential for reducing capital reserves, up to 15% in certain projections, is likely to be a notable outcome of this maturation. This reduction in capital reserves positions quantum computing to deliver an operating income of $40 billion to $70 billion to banks and other financial services companies.[10]

Automotive

According to market research company MarketsandMarkets, the quantum computing in automotive market is expected to reach $143 million in 2026 and to $5.2 billion in 2035.[12] Even a modest increase in productivity (2%–5%) within an industry that spends $500 billion annually on manufacturing costs could yield an annual value of $10 billion to $25 billion.[4]

In this industry, quantum computing has the potential to revolutionize R&D, product design, supply-chain management, production, and the optimization of mobility and traffic control. As an illustration, this technology could be utilized to reduce manufacturing costs associated with the process and expedite production cycles. It achieves this by optimizing aspects like path planning within intricate multirobot operations, such as welding, gluing, and painting.[4]

Quantum computing can also have other positive effects such as reduction of defects. For instance, PASQAL and BMW are partnering to use quantum algorithms for simulating the formation of metallic pieces, with the goal of identifying defects and ensuring parts meet specifications.[6]

This sector will also benefit from initiatives being taken in the manufacturing sector to improve supply chain performance and enhance maintenance optimization. In a partnership between Multiverse and German multinational engineering and technology company Bosch, quantum algorithms are being utilized to predict and detect defects in production lines, addressing the challenge of managing extensive data for accurate predictions. Their aim is to establish digital twins of factory lines to predict supply chain failures and optimize maintenance.[6]

CONCLUSION

The quantum computing industry is currently in the NISQ era, marked by substantial error rates and the constrained size of quantum processors, which significantly hampers the effectiveness of quantum computers. Despite the current infancy stage of quantum computing, its effects are anticipated to increase with ongoing application development. Private investment in this innovation has seen a notable increase in recent years, accompanied by significant allocations of public funds aimed at propelling its growth. The substantial financial investments dedicated to quantum computing R&D have spurred recent progress in both quantum computing hardware and software, along with the development of innovative error mitigation and suppression methods. Certainly, all this activity doesn't automatically equate to immediate commercial success. Nonetheless,

with advancements in this innovation, several pathways for monetization are likely to become available.

The aforementioned discussion has shown that there are numerous underlying value creation mechanisms and monetization models of quantum computing. A number of trials and tests carried out so far indicate that this innovation has the potential to reshape various business functions, including R&D, supply chain management, and production. Quantum computing can also greatly improve the ability to attract and retain customers. While most organizations will not be in a position to buy a quantum computer in the near term, emerging business models such as QCaaS allow them to access and monetize this innovation faster and better.

Overall, while quantum computers are not yet ready for widespread industry use, startups are actively identifying potential applications for the technology. Quantum computing thus holds promise for delivering positive outcomes for businesses, economies, and societies.

REFERENCES

1. F. Bova, A. Goldfarb, and R. Melko, "The business case for quantum computing," *MIT Sloan Manage. Rev.*, Mar. 7, 2023. Accessed: Nov. 15, 2023. [Online]. Available: https://sloanreview.mit.edu/article/the-business-case-for-quantum-computing/
2. "Betting big on quantum," McKinsey & Company, New York, NY, USA, Sep. 13, 2022. [Online]. Available: https://www.mckinsey.com/featured-insights/sustainable-inclusive-growth/chart-of-the-day/betting-big-on-quantum
3. M. Brooks, "Quantum computers: What are they good for? For now, absolutely nothing. but researchers and firms are optimistic about the applications," *Nature*, May 24, 2023. Accessed: Nov. 15, 2023. [Online]. Available: https://www.nature.com/articles/d41586-023-01692-9
4. M. Biondi et al., "Quantum computing use cases are getting real—What you need to know," McKinsey & Company, New York, NY, USA, Dec. 14, 2021. [Online]. Available: https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/quantum-computing-use-cases-are-getting-real-what-you-need-to-know
5. M. Shirer, and H. West. "IDC forecasts worldwide quantum computing market to grow to $7.6 billion in 2027." International Data Corporation. Accessed: Nov. 15, 2023. [Online]. Available: https://www.idc.com/getdoc.jsp?containerId=prUS51160823
6. D. Leprince-Ringue, "7 ways that quantum computing is making an impact in the real world," Sifted, London, U.K., Aug. 10, 2023. [Online]. Available: https://sifted.eu/articles/seven-applications-quantum-computing
7. J. Ruane, A. McAfee, and W. D. Oliver, "Quantum computing for business leaders: Will the reality live up to the hype?" *Harvard Bus. Rev.*, Jan./Feb. 2022. Accessed: Nov. 15, 2023. [Online]. Available: https://hbr.org/2022/01/quantum-computing-for-business-leaders
8. R. Nagao, "Japan to offer $30m in aid for quantum computer sharing in industry, finance," *Nikkei*, Apr. 14, 2023. Accessed: Nov. 15, 2023. [Online]. Available: https://asia.nikkei.com/Business/Technology/Japan-to-offer-30m-in-aid-for-quantum-computer-sharing-in-industry-finance
9. "Quantum computing in healthcare: A billion-dollar revolution by 2030," GlobeNewswire, Los Angeles. CA, USA, Sep. 7 2023. [Online]. Available: https://www.globenewswire.com/news-release/2023/09/07/2739043/28124/en/Quantum-Computing-in-Healthcare-A-Billion-Dollar-Revolution-by-2030.html
10. M. Langione, C. Tillemann-Dick, A. Kumar, and V. Taneja, "Where will quantum computers create value—And when?" Boston Consulting Group, Boston, MA, USA, May 13, 2019. [Online]. Available: https://www.bcg.com/publications/2019/quantum-computers-create-value-when
11. F. F. Bova, A. Goldfarb, and R. Melko, "Quantum computing is coming. What can it do?" *Harvard Bus. Rev.*, Jul. 16, 2021. Accessed: Nov. 15, 2023. [Online]. Available: https://hbr.org/2021/07/quantum-computing-is-coming-what-can-it-do
12. "Quantum computing in automotive market worth $5,203 million by 2035," GlobeNewswire, Los Angeles. CA, USA, Sep. 26, 2023. [Online]. Available: https://www.globenewswire.com/news-release/2023/09/26/2749879/0/en/Quantum-Computing-in-Automotive-Market-worth-5-203-million-by-2035-Exclusive-Report-by-MarketsandMarkets.html

NIR KSHETRI is a professor in the Bryan School of Business and Economics, the University of North Carolina at Greensboro, Greensboro, NC, 27412, USA, and *IT Professional's* IT Economics editor. Contact him at nbkshetr@uncg.edu

EDITORS: Anne Elster, anne.elster@gmail.com
Sonia Lopez Alarcon, slaeec@rit.edu

This article originally appeared in
Computing in Science & Engineering
vol. 24, no. 6, 2022

DEPARTMENT: NOVEL ARCHITECTURES

Quantum Computing and High-Performance Computing: Compilation Stack Similarities

Sonia Lopez Alarcon, *Rochester Institute of Technology, Rochester, NY, 14623, USA*
Anne C. Elster, *Norwegian University of Science and Technology, NO-7491, Trondheim, Norway*

There is a great deal of focus on how quantum computing as an accelerator differs from other traditional high-performance computing (HPC) resources, including accelerators like GPUs and field-programmable gate arrays. In classical computing, how to design the interfaces that connect the different layers of the software stack, from the applications and high-level programming language description, through compilers and schedulers, and down to the hardware and gate level, has been critical. Likewise, quantum computing's interfaces enable access to quantum technology as a viable accelerator. From the ideation of the quantum application to the manipulation of the quantum chip, each interface has its challenges. In this article, we discuss the structure of this set of quantum interfaces, their many similarities to the traditional HPC compilation stack, and how these interfaces impact the potential of quantum computers as HPC accelerators.

Quantum computing will not replace classical high-performance computing (HPC) systems—at least not in the foreseeable future. However, there is currently a lot of research focusing on how they can be used as an accelerator for quantum simulations, machine learning applications,[1,2] optimization and combinatorial problems,[3,4] and other computationally expensive applications.[5] Quantum computing grew from the birth of quantum information theory in 1970 and Benioff's four publications in the early 1980s that showed, for the first time, how quantum computers were theoretically possible.[a] The first experimental quantum gates were implemented shortly after. IBM, Intel, Google, IonQ, Honeywell, Xanadu, and many other large companies and start-ups are now all investing in advancing this technology, to the point that it is hard to keep up with the number of research papers being published.

The power of quantum computing stems from how densely it can represent information. This comes from the quantum superposition property—the linear combinations of two or more states, much like a combination of musical tones results in a new unique sound—and entanglement—the inexplicable correlations that happen between quantum bits (qubits). Interference is used to cancel portions of the superposition, similar to the use of noise-canceling technologies in headphones. In addition, quantum gates are reversible, which means that the system preserves the information at any point of the execution. The theory says that, since information is not destroyed, application of the operands (also known as quantum gates) does not consume power. However, note that power is required to generate the operands and to keep a closed quantum state.

While traditional computing systems store zeros and ones, a two-qubit system has been claimed by IBM[b] to store the equivalent entangled state information of 512 classical bits, 10 qubits (the equivalent of

[a]https://www.anl.gov/article/remembering-paul-benioff-renowned-scientist-and-quantum-computing-pioneer

[b]https://www.ibm.com/thought-leadership/institute-business-value/report/quantum-decade

1521-9615 © 2023 IEEE
Digital Object Identifier 10.1109/MCSE.2023.3269645
Date of current version 1 June 2023.

16 kB of classical bits), and the current large quantum computing systems with 100 and 280 qubits would need a number of bits equal to the number of atoms on planet Earth and the universe, respectively.

Quantum states are extremely delicate, and the challenge is to keep a system of qubits in its superposed and entangled state and manipulate them in a controlled way. External interactions in all energy forms, even at the smallest scale, can easily make the state of the quantum system *fall out of coherence*, inducing noise as an error. This is a major challenge for implementing concrete quantum computers. Superconducting qubits need to be maintained at temperatures as cold as or colder than outer space to not be susceptible to such errors. Recent technologies using photonic and diamond-defect designs try to overcome this. In the photonic case, the sensors still need to be supercooled, whereas systems based on diamond defects currently are limited to single-digit qubits.[c]

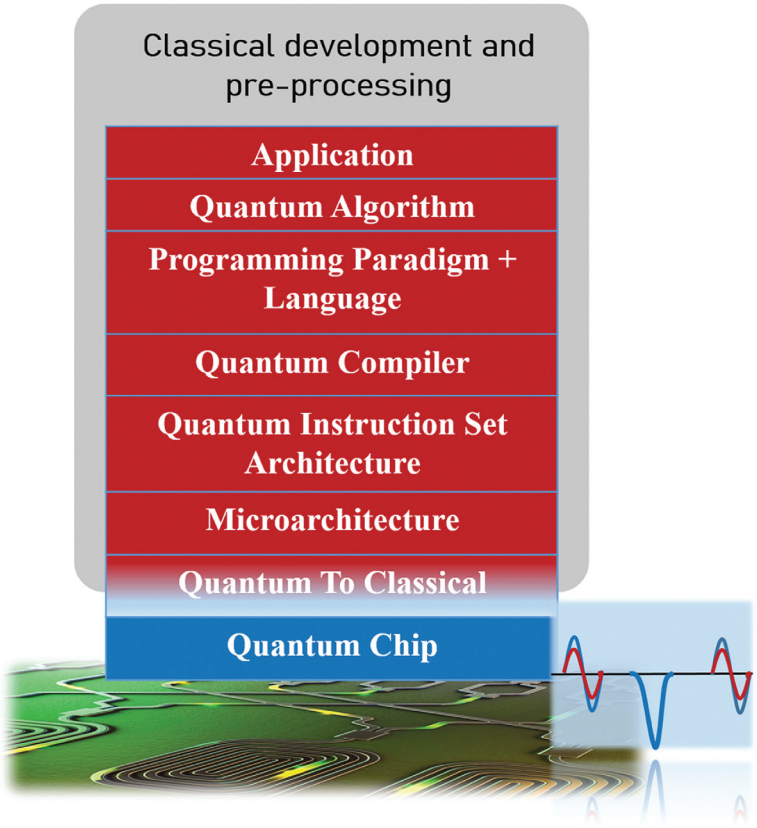


FIGURE 1. The quantum computing software stack can also be envisioned as this stack of interfaces.

The current technology used in quantum computing is known as the noisy intermediate-scale quantum (NISQ) system, a term that was coined by John Preskill.[6] Small numbers of qubits with high error rates and limited connectivity define these systems. On these systems, only very specific applications that are hardly considered useful can outperform classical implementations. This is, however, a necessary step toward powerful quantum computing, with a high enough number of qubits to allow not only computational power but also tolerance to error.

The progress is real, mainly and most crucially at the technology level, but also in all of the other layers that separate the user from the physical quantum system: algorithms, applications, programming models, and compilers.[7] Each of these layers is an *interface* that abstracts out the details of the layers below and simplifies the development task.

An IEEE *Computing in Science & Engineering* "Leadership Computing" department recently discussed the integration of quantum computing and HPC in a single software stack.[8] This time, we take a closer look at the software stack that bridges the gap between the quantum application and the actual quantum systems leveraging quantum mechanical properties.

In classical computing, interfaces support the development steps, from the high-level programming language description of an application to controlling electrons through semiconductor transistors. Similarly, quantum computing relies on a set of steps and interfaces.[7] The actual computation on the quantum hardware is only the final step, while the majority of the development and preprocessing (D&PP) is done classically, with a quantum mindset, as shown in Figure 1.

The challenges of this D&PP are in no way negligible. The compiler, in particular, calls for a series of optimizations and graph problems that threaten the scalability of quantum computers.

QUANTUM COMPUTING INTERFACES

A quantum computer is a quantum system that evolves according to quantum mechanical principles from an

[c]Published 20 March 2023: https://www.eetimes.eu/the-status-of-room-temperature-quantum-computers/

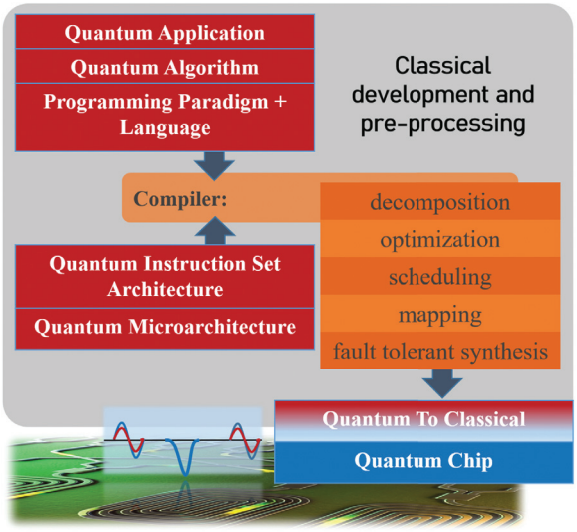


FIGURE 2. Quantum compilation takes in information from different interfaces. Through a number of steps, it generates the fault-tolerant synthesis of the quantum applications.

initial state to a final state. If all things go well, this final state contains the solution to a computational problem.

A stack of interfaces makes it possible to take a quantum application from ideation to reality. The software stack and the interfaces needed to realize the quantum application are envisioned as a workflow in which the compiler plays a central role, as illustrated in Figure 2.

A potential quantum application, e.g., identifying the maximum clique of a graph, makes use of a quantum algorithm, e.g., Grover's algorithm. The "only" quantum aspect of these two interfaces is the understanding of core and fundamental quantum properties and how classical problems can be framed in a quantum context. This understanding is probably the greatest gap keeping the general scientific community from taking advantage of quantum acceleration at this point in time, even greater than the technology itself. It is still unclear which applications can be efficiently accelerated by quantum means.

Many scientists (mainly theoretical physicists) have described or envisioned these applications and algorithms as mathematical exercises with "pencil and paper." However, to take these closer to an actual quantum implementation, a description on a suitable programming language, e.g., quantum-specific Python extensions, should define the steps of the application and its quantum algorithms in a way that can then be compiled targeting a specific quantum instruction set architecture and microarchitecture.

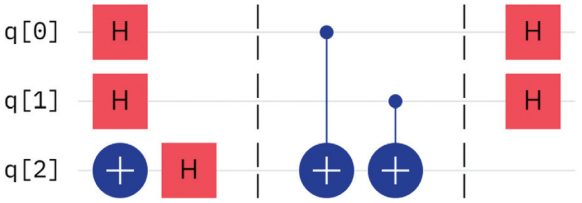


FIGURE 3. An example of a basic circuit, including a three qubit register, Hadamard (H), controlled NOT and NOT (+) gates built with IBM's Quantum Composer.[9] The computation proceeds from left to right, from the initial to final quantum state of the three-qubit register.

The quantum instruction set architectures in their current form are far from being a set of general-purpose instructions but, rather, are a sort of single- or two-qubit basic operand known as gates. Figure 3 depicts a basic circuit with several single- and two-qubit gates operating on a three-qubit register. Each quantum computing technology has its own set of native gates. The implementation of these quantum gates is in the form of analog signals (microwaves, lasers, or others, depending on the technology) that act on the qubits and that are generated and controlled by the microarchitecture. Also, the qubits' connectivity map is part of the system's architecture since not all qubits can interact with all other qubits.

Therefore, with a description of the quantum application, typically as a collection of quantum gates known as a quantum circuit, as well as the quantum architecture and microarchitecture's information, the quantum compiler can generate the necessary information to control the quantum system: its initialization and quantum time evolution to the final quantum state on the quantum chip (Figure 2).

Notice that the D&PP down to the classical quantum interface does not involve any physical quantum interaction. Instead, all of this is done on classical computing systems.

QUANTUM COMPILATION

Compilation for a quantum computer system involves a number of steps that take the high-level language description all the way down to generating the control signals. The decomposition of the high-level gates breaks them down into the native gates of the architecture. For instance, Figure 5 represents a swap gate and its equivalent decomposition into three controlled NOT (CNOT) gates. Sometimes, consecutive quantum gates cancel each other or are commutative in the order of the execution. These optimizations are taken

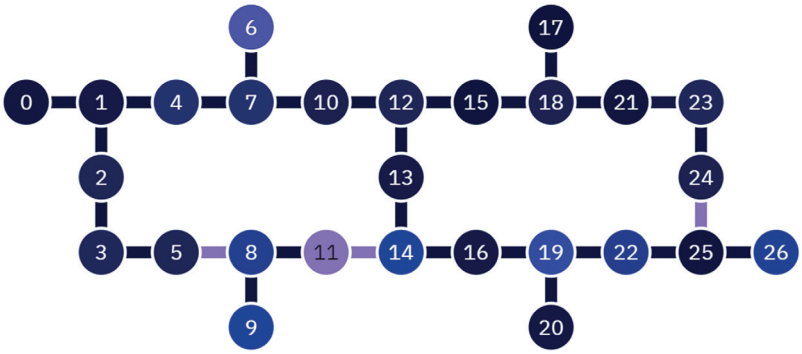


FIGURE 4. IBM Quantum's 27-qubit Kolkata back end. The shades of color represent the quality of the qubits and the links among them. (A darker shade means better quality.) This quality is based on different noise and error metrics. Different back ends have different maps and noise levels.[10]

care of before scheduling the order in which the operands will take place, respecting all dependencies and exploiting parallelism when possible.

The mapping stage involves two operations: mapping and routing. In the quantum circuit high-level description, gates act on quantum variables that we call logical qubits. These have to be mapped to physical qubits on the architecture's map. Then, as the execution evolves, the qubits that need to interact with each other through two-qubit gates are routed to physically adjacent qubits in the architecture's map.

For example, if logical qubit 1 (Q1) and logical qubit 2 (Q2) in Figure 5 were mapped originally to physical qubits 6 and 10 in Kolkata's map (Figure 4), Q1 would have to be rerouted to qubit 7, so it would be physically adjacent to 10. Qubit rerouting is done adding swap gates. Low connectivity is a critical problem in the current NISQ systems,[6] with high noise levels, low coherence times, and no error-correction protocols enabled yet. In IBM's superconducting systems, the necessary swap gates (three CNOT gates each) for qubit routing do, on the other hand, accumulate link error and increase the depth of the circuit in ways that often surpass the quantum coherence time of the system, resulting in too noisy of an output to be useful. Efficient routing algorithms route not only to ensure correctness but also to minimize noise.

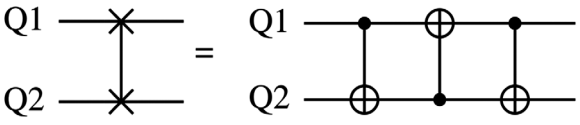


FIGURE 5. SWAP gate and its decomposition in three CNOT gates.

Once the final circuit is built with all of its optimizations and added swaps and the gates are scheduled, the compiler will generate the fault-tolerant synthesis according to the system's microarchitecture.

Scheduling and mapping/routing are well-known NP-hard problems that can be found in a plethora of other fields. The HPC community is well aware of the time complexity, memory, and hardware resources required to solve these problems. Inputs are large graphs, such as dependency graphs or the connectivity map. The goal usually involves an optimization problem in which time, hardware usage, error, or noise need to be minimized.

Quantum Versus Classical Compilation

Quantum and classical compilation processes have some analogies: starting from the dependency graphs, the scheduling of operations and allocation of resources need to be performed. Those with experience in field-programmable gate array (FPGA) acceleration may notice the resemblance with the high-level synthesis

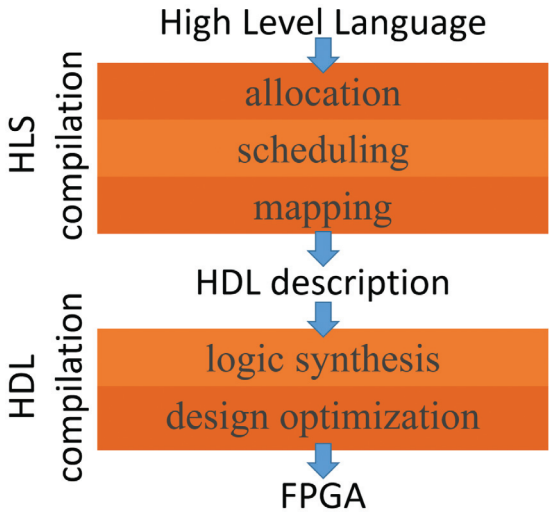


FIGURE 6. High-level synthesis (HLS) full compilation process: from a high-level language description, an implementation in a hardware description language (HDL) is generated (Verilog or VHDL) to then be synthesized and run on the field-programmable gate array (FPGA).

full compilation stack. Figure 6 represents this stack, from the high-level language description to the FPGA execution.

Similar to the decomposition stage, allocation identifies the basic elements needed to implement the operations described in high-level language. Then, operations are scheduled and mapped (also known as bound) to those elements. From the hardware description language (HDL), the logic synthesis and necessary optimizations generate the files of zeros and ones that will take care of the actual implementation on an FPGA.

FPGA users may have experience with the limitations of this process: compilation on the fly is prohibitive in terms of time; the high-level language description of the computations cannot implement recursive calls and may have issues with pointers and memory accesses; and, in summary, hardware design skills that are not accessible to all users are necessary. Whether using HDLs or high-level synthesis, the implementation's description is hardware description, not software.

QUANTUM DESCRIPTION LANGUAGES

Setting aside the necessary understanding of quantum algorithms to be able to develop quantum applications, the quantum programming paradigm is the most immediate interface to the quantum computer user. The lexicon used around the description of quantum applications seems to indicate that we are describing hardware, just like VHDL or Verilog are used to describe FPGA hardware: first of all, because we call the description a "quantum circuit"; second, because the operands are referred to as "gates"; and last, because the operations are described at a qubit-by-qubit granularity. There are no memory, no data types, and no general-purpose flexible instructions. None of this is compatible with software execution.

However, if we take the term hardware as its literal translation of something that is hardwired or physically palpable, this term does not apply either. Figure 7 gives an example of what really happens on the execution of a quantum application, after all of the steps in Figure 2 have been completed, and the quantum system finally kicks in. The quantum chip contains the "hardwired" qubits. The quantum D&PP have generated the sets of gates as analog control signals. Groups of these control signals arrive at the quantum chip to act on the qubits and alter the state of the quantum system. If, like in Grover's algorithm, the sets of gates need to be iterated through multiple times, the corresponding control signals can just be repeated as many times as needed. They are not real "hardwired" gates.

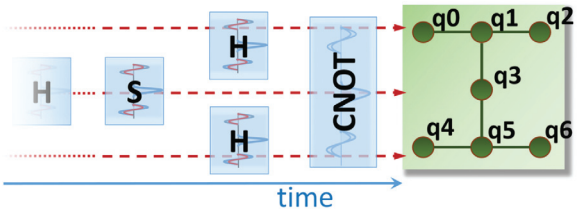


FIGURE 7. Quantum Hadamard (H), Phase (S) and Controlled-NOT (CNOT) are sent to the quantum chip as control signals.

The quantum programming models leverage high-level languages, Python most commonly, to describe these extremely fine-grained computations. The state of the quantum system contains the information. The quantum chip and its qubits act, actually, as a very short-lived memory upon which we need to operate before the system falls out of its delicate quantum equilibrium.

Although the quantum application is described using high-level languages, such as Python, much has to happen at the development level to give "quantum

APPENDIX: RELATED ARTICLES

A1. *The Quantum Decade*, 3rd ed., IBM, Armonk, NY, USA. [Online]. Available: https://www.ibm.com/thought-leadership/institute-business-value/report/quantum-decade

A2. "The status of room-temperature quantum computers," *EE Times Europe*, Mar. 2023. [Online]. Available: https://www.eetimes.eu/the-status-of-room-temperature-quantum-computers/

A3. F. G. Fuchs, V. Falch, and C. Johnsen, "Quantum Poker—A game for quantum computers suitable for benchmarking error mitigation techniques on NISQ devices," *Eur. Physical J. Plus*, vol. 135, no. 4, Apr. 2020, Art. no. 353, doi: 10.1140/epjp/s13360-020-00360-5.

A4. "Quantum computing is the future, and schools need to catch up: Top universities are finally bringing the excitement of the quantum future into the classroom," *Scientific Amer.*, Mar. 2023. [Online]. Available: https://www.scientificamerican.com/article/quantum-computing-is-the-future-and-schools-need-to-catch-up/

description languages" a software-like feeling. This includes the use of more generic application programming interfaces (APIs) and libraries of computations. The implementation of reasonably long-lasting quantum memory is also a key piece that is missing in this picture and that currently forces the start of every computation to long strings of operands just to initialize the states to the data that are going to be operated on.

CONCLUSION

Despite the many challenges, quantum computing holds real potential for accelerating certain applications, such as machine learning applications, combinatorial and optimization problems, and quantum molecular simulation. Proof of quantum acceleration for practical, real-world cases will most likely have to wait until the post-NISQ era is reached, with a higher number of qubits and the inclusion of error-correction protocols. Most companies are looking at a five-year timeline,[11] primarily depending on technology advances.

Meanwhile, a solid stack of interfaces needs to be developed to support these future applications. In this article, we discussed how the D&PP of quantum applications entails a series of classical steps that can quickly become unmanageable, even more so given that the number of qubits and quantum gates required often can grow exponentially with the size of the problem—without even considering error-correction mechanisms. The good news is that these are not new problems. Scheduling, mapping, routing, allocation, and optimization problems are common in other fields, and heuristics can be applied to approximately solve these problems more efficiently. An efficient programming paradigm is yet to be defined, but the field can leverage the decades of experience of the HPC community at creating interfaces that bridge knowledge gaps and that conceal the intricacies and challenges of the physical implementation.

ACKNOWLEDGMENTS

The second author would like to thank the Center for Geophysical Forecasting at NTNU and RCN (NFR Project 309960) for its support during the preparation of this article.

REFERENCES

1. S. Lopez Alarcon, C. Merkel, M. Hoffnagle, S. Ly, and A. Pozas-Kerstjens, "Accelerating the training of single-layer binary neural networks using the HHL quantum algorithm," 2022. [Online]. Available: https://arxiv.org/abs/2210.12707
2. E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," 2014, *arXiv:1411.4028*.
3. L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin, "Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices," *Physical Rev. X*, vol. 10, no. 2, Jun. 2020, Art. no. 021067, doi: 10.1103/PhysRevX.10.021067.
4. A. Haverly and S. Lopez Alarcon. "A comparison of quantum algorithms for the maximum clique problem." Medium. Accessed: May 7, 2023. [Online]. Available: https://medium.com/xanaduai/a-comparison-of-quantum-algorithms-for-the-maximum-clique-problem-4cd8984cea59
5. F. G. Fuchs, K. O. Lye, H. Møll Nilsen, A. J. Stasik, and G. Sartor, "Constraint preserving mixers for the quantum approximate optimization algorithm," *Algorithms*, vol. 15, no. 6, Jun. 2022, Art. no. 202, doi: 10.3390/a15060202.
6. J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, Aug. 2018, Art. no. 79, doi: 10.22331/q-2018-08-06-79.
7. L. Riesebos et al., "Quantum accelerated computer architectures," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2019, pp. 1–4, doi: 10.1109/ISCAS.2019.8702488.
8. M. Schulz, M. Ruefenacht, D. Kranzlmuller, and L. Schulz, "Accelerating HPC with quantum computing: It is a software challenge too," *Comput. Sci. Eng.*, vol. 24, no. 4, pp. 60–64, Jul./Aug. 2022, doi: 10.1109/MCSE.2022.3221845.
9. "IBM quantum composer." IBM Quantum. Accessed: May 7, 2023. [Online]. Available: https://quantum-computing.ibm.com/composer/docs/iqx/
10. "Compute resources." IBM Quantum. Accessed: May 7, 2023. [Online]. Available: https://quantum-computing.ibm.com/services/resources
11. "IBM's roadmap for scaling quantum technology." IBM. Accessed: May 7, 2023. [Online]. Available: https://www.ibm.com/blog/

SONIA LOPEZ ALARCON is an associate professor in computer engineering at Rochester Institute of Technology, Rochester, NY, 14623, USA. Contact her at slaeec@rit.edu.

ANNE C. ELSTER is a professor in computer science at the Norwegian University of Science and Technology in Trondheim, NO-7491, Norway. Contact her at anne.elster@gmail.com.

EDITORS: Kathryn Mohror, mohror1@llnl.gov
John M. Shalf, JShalf@lbl.gov

This article originally appeared in
Computing in Science & Engineering
vol. 24, no. 4, 2022

DEPARTMENT: LEADERSHIP COMPUTING

Accelerating HPC With Quantum Computing: It Is a Software Challenge Too

Martin Schulz, *Technical University of Munich, 80333, Munich, Germany*

Martin Ruefenacht, *Leibniz Supercomputing Centre of the Bavarian Academy of Sciences and Humanities, 85748, Garching near Munich, Germany*

Dieter Kranzlmüller, *Ludwig–Maximilians–Universität München, 80538, Munich, Germany*

Laura Brandon Schulz, *Leibniz Supercomputing Centre of the Bavarian Academy of Sciences and Humanities, 85748, Garching near Munich, Germany*

With quantum computing (QC) maturing, high-performance computing (HPC) centers are already preparing to host early-phase production versions of such systems. Unlike their experimental predecessors in physics laboratories, with a very small and dedicated user community, this next generation of systems needs to serve a wider user community and must work in concert with existing HPC systems and software stacks. This article describes our vision for an integrated ecosystem that combines existing HPC and evolving quantum software stacks into a single system to enable a common and continuous user experience. This integration comes with several major challenges as quantum systems pose significantly different requirements including increased need for compilation at run time, long optimization times, statistical evaluations of results, and the need to work with few centralized resources. To overcome these challenges, new scheduling approaches on the HPC side and new programming approaches on the QC side are required.

MOTIVATION

Quantum computing (QC), i.e., the idea of using quantum states and transformations to express computation, is taking shape. After decades of experimentation in physics laboratories, many large-scale research efforts in academia, laboratories, and industry world-wide have started to target usable and accessible quantum computing devices. These efforts explore a wide range of underlying technologies from superconducting qubits, spin-qubits, ion traps to neutral atoms, to name just a few.

While these developments are highly promising, it is becoming clear that quantum computing systems will not replace existing compute architectures; they more likely augment them by accelerating certain suitable tasks or kernels. The computation of other kernels and work needed for I/O and workflow management will (at least for the foreseeable future) remain bound to the existing compute approaches. Additionally, quantum computing relies on several compute-intensive tasks, which require support from HPC systems. Consequently, QC must seamlessly become part of HPC, enabling common user access and experience.

In order to enable the needed integration, it will be essential to not only develop QC hardware and physically connect it to HPC systems, which is currently the main focus for several groups with approaches ranging from loosely coupled cloud or modular access,[1,2] to near quantum compute options,[3] to actual deep integration targeting low-latency access.[4] We also need to focus on a continuous software stack that enables a user to harness the combined computational capabilities of both

1521-9615 © 2023 IEEE
Digital Object Identifier 10.1109/MCSE.2022.3221845
Date of current version 4 January 2023.

system components in a seamless fashion. In this article, we describe our efforts toward this combined software stack: we build on novel features for the HPC software side, particularly dynamic adaptivity and malleability across all software components and modern workflow features. On the QC side, we start with the standard programming packages available to QC researchers, such as Cirq[5] and Qiskit,[6] and link them to the HPC resources management as well as augment their back-ends to efficiently and transparently onload compute intensive tasks back to the HPC system, ideally using the HPC workload manager to identify otherwise unused resources.

Combining these two worlds, however, is not trivial, as they work on radically different assumptions. The HPC world is trimmed for low latency and precompilation; on the other hand, QC systems require a much larger runtime component. The latter is caused by compiling at runtime as input arguments are likely to change the intended quantum program (i.e., the generated circuits). Further, the computation needed to compile and map to the final topology of the QC system is substantial.

Also, QC systems only target single-user operations and offer limited support for multiuser operations. Therefore, integration requires a careful redesign of the runtime component to enable efficient overlap of QC executions from multiple users and to mitigate idle times caused by complex preparation tasks.

Our framework addresses these challenges by adding the needed resource isolation coupled with QC operation interleaving and integrates QC execution scheduling with the ability to onload work needed to drive the QC execution back to the HPC system efficiently. Our approach bridges the software stacks from the two worlds, enabling a single system abstraction for the end user while ensuring high system efficiency.

We are implementing our vision as part of a series of European HPC Software Stack projects and with the Munich Quantum Valley (MQV), a large research initiative driving the development of three different quantum computing technologies with a common software stack. The resulting software will ultimately drive the usage of QC technologies at the Leibniz Supercomputing Centre (LRZ) and compute centers world-wide.

THE ANATOMY OF HPC VERSUS QC SOFTWARE STACKS

Seamless integration needs to support both the HPC users requiring a traditional HPC software stack and the QC users accustomed to the existing stand-alone QC development environments, as well as emerging user groups targeting hybrid HPCQC operations with direct access to QC systems from within HPC systems. Consequently, we need to, at a minimum, support the existing HPC and QC software stacks while also adding support for integrated usage.

A key challenge comes from existing software stacks; the two types of systems are radically different in structure, approach, and user interfaces. This disparity is rooted in the maturity and current scale of the different system components and the diverging requirements stemming from the fundamentally different technologies. Consequently, simply integrating one stack into the other is neither possible nor desirable. Instead, we will build on state-of-the-art software stacks for HPC and QC, respectively, and work to provide an efficient link between the stacks offering the needed integration. This will ensure a solution that offers the best of both worlds while offering a cohesive, integrated view of the system for the users and developers.

HPC Software Stacks

On the HPC side, we build on top of widely available technology forming the software stacks as they are now available on most HPC systems. Examples for this are HPC enabled stacks, such as OpenHPC,[a] the E4S initiative,[b] or the DEEP-SEA stack developed for the European exascale systems,[c] as they are used in most large-scale HPC centers. These include state-of-the-art components for compilers, runtimes, parallel programming abstractions including MPI[7] and OpenMP,[8] support for accelerators following the evolution of system architectures, as well as a wide range of libraries to support efficient application development.

HPC stacks typically target the optimization of execution time by implementing as much work as possible at compile time (compilation, optimization, job preparation, etc.) and, with that, reducing runtime overheads. The latter is then limited to scheduling operations at the job level and actual execution overheads. For this reason, compiled languages such as C, C++, or Fortran are dominant, while interpreted languages play a minor role, e.g., only for coarse-grained workflow orchestration. Further, HPC software stacks typically offer system access via command-line driven batch execution, enabling users direct and fine-grained control of their execution.

QC Software Stacks

Quantum computing stacks, on the other hand, feature a radically different approach to programming

[a]https://openhpc.community/
[b]https://e4s-project.github.io/
[c]https://www.deep-projects.eu/

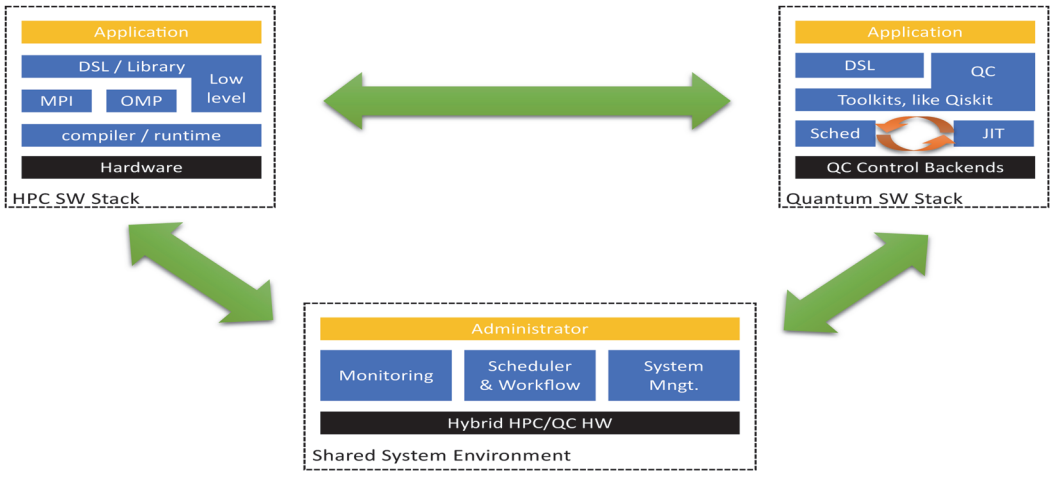


FIGURE 1. Integrating the HPC and QC software environments as well as the overall system environments to reach a seamless hybrid HPCQC system with a combined workflow.

and computing. QC programs, typically in the form of quantum circuits, are defined at runtime using interpreted languages, such as Python, for convenience. Typical programming front-ends are frameworks such as Qiskit[6] or Cirq,[5] which use Python as their base. Programs are then assembled and translated into lower level representations at runtime. The latter is driven by the property of QC programs that changing parameters influences the structure and composition of the program and hence requires new compilations.

This approach works for current scenarios, where individual users access a specific quantum system to execute a particular circuit. As circuits are small, usage is typically interactive and preparation time for a particular problem is ignored as it does not contribute to overall execution time. This, however, changes when users intend to run larger as well as multiple QC applications with only minor parameter changes triggering frequent recompilation and optimization. To maximize system utilization, the resource QC system, which is rare compared to the typical abundance of HPC nodes, needs to be time and space shared; this prohibits the currently used single-user allocation model.

Gaps and Challenges

To combine these two software stacks with their radically different properties, we must include the inherent runtime component of the QC stack and attempt to hide it as much as possible during the execution. This requires both novel scheduling techniques to overlap QC circuit compilation and execution, to overlap executions from multiple users, and novel approaches to reduce the QC circuit generation and optimization times by parallelizing and onloading this work back to the HPC system.

Additionally, it will be important to unify the system software environments, especially with respect to system management, monitoring, and scheduling in order to enable an efficient and stable operation of the joint HPCQC system. This requires new developments to fuse the monitoring environments available in the different systems, an integrated scheduling approach combining the coarse-grained batch scheduling with the fine-grained scheduling of individual QC experiments, and a joint management concept for the involved nodes or subsystems. This will ensure that system administrators can monitor and manage the overall system as one entity and using a single set of consistent policies.

LINKING THE HPC AND QC SOFTWARE STACKS

As discussed in the previous section, we need to build on top of the existing software stacks for both HPC and QC, to ensure continuity for users, but at the same time we also need to provide an integrating bridge. This concept is illustrated in Figure 1, showing the interactions of the two software stacks for HPC and QC, and the connection to the underlying shared system environment. In the following, we will discuss the interactions between the components in more details.

Support for Offloading

The most natural way of integration is to allow HPC systems to push workloads to the QC system, a mechanism we refer to as "offloading" in congruence with offloading mechanisms in other accelerators. This requires the ability to specify quantum computation from within HPC programs, which is challenging due

to the different programming approaches. To solve this challenge, we are developing new techniques to 1) enable Python-driven front-ends and integrate them into existing offload approaches such as OpenMP and 2) use noninterpreted approaches directly in C/C++, similar to the XACC approach.[9]

In all cases, the quantum environments require a compilation on the fly, i.e., at runtime, as input parameters will (in most cases) affect the required circuit, as initial conditions need to be directly encoded. Such a compilation can be trivial in the naive case. However, in most cases, it requires intensive computations to achieve the needed optimizations and mapping to the underlying qubit topologies on the quantum devices. This introduces extra latencies, which will block the calling entity. While this is acceptable for individual experiments through direct access from frameworks like Qiskit, it causes substantial delays when used from within HPC jobs. In the latter case, a large number of nodes are held until the translation and optimization is complete, causing unacceptable idle times.

Support for Onloading

In order to combat the challenges identified above, we need to find ways to speed up compilations and optimization of quantum circuits. Using the connected HPC resources is a promising approach but requires 1) the ability to send optimization requests back to the HPC system and 2) novel quantum circuit optimizers, which are parallelized. The latter requires quantum development environments to be implemented on compiled platforms with the necessary parallel structures. We are working on new compilers and optimizers that enable such optimization. Our current targets are implementations in Rust, which combine safe coding practices and efficient threading models for parallelization.

Common Scheduling Mechanisms

A common issue for both onloading versus offloading is efficient scheduling: when to execute which QC computation, how to schedule compilation steps versus executing on the actual QC system, and how to onload QC operational considerations back to the HPC systems. Such scheduling needs to be resource-driven, i.e., the execution from multiple users must be interleaved and compilation of quantum circuits must be pushed back to the HPC system looking for idle resources, e.g., for nodes that actually issued the quantum computation and may be idle until the results return.

CONCLUSIONS

Quantum computing is a highly promising technology that has the potential to accelerate certain computations substantially. However, QC must be integrated into the context of existing HPC ecosystems, and for this we need to carefully consider requirements and changes to the respective software stacks, too.

This article presents our vision for a combined HPC and QC software stack. It builds on existing software stacks from HPC and QC, and extends them to ensure tight integration. We provide extensions to include scheduling components for QC-enabled portions on the HPC side. At the same time, on the QC side, we add programming models and the ability to push operations to the HPC side to speed up operations. For this, we leverage recent developments in the HPC software stack for malleability to integrate the more dynamic nature of QC computations.

The result is an efficient link between the HPC and QC software stacks, allowing them to leverage each other and achieve efficient hybrid execution. Ultimately, this will form the needed bridge between the two worlds and enable efficient HPCQC operations. We are pursuing this direction both as part of several European software stack projects and the MQV with the goal of providing the first truly integrated HPCQC software stack to be deployed in production.

ACKNOWLEDGMENTS

This work was supported in part by the German Federal Ministry for Education and Research under Grant 13N15689 (DAQC), Grant 13N16063 (Q-Exa), Grant 13N16188 (MUNIQC-SC), and Grant 13N16078 (MUNIQC-ATOMS), in part by the German Federal Ministry for Economic Affairs and Climate Action under Grant 01MQ22004 C (QuaST), as well as the Bavarian State Ministry of Science and the Arts as part of Munich Quantum Valley (MQV).

REFERENCES

1. V. Bartsch et al., "QC | HPC: Quantum for HPC," Oct. 2021, doi: 10.5281/zenodo.5555960.
2. D. Binosi et al., "European quantum computing & simulation infrastructure," 2022. [Online]. Available: https://qt.eu//app/uploads/2022/02/20220202_HPC-QCS-JWP-final.pdf
3. M. P. Johansson, E. Krishnasamy, N. Meyer, and C. Piechurski, "Quantum computing–a European perspective," Sep. 2021, doi: 10.5281/zenodo.5547408.
4. M. Ruefenacht et al., "Bringing quantum acceleration to supercomputers," 2022. [Online]. Available: https://meetiqm.com/uploads/documents/IQM_HPC-QC-Integration-Whitepaper.pdf

5. C. Developers, "Cirq," Apr. 2022, See full list of authors on Github: https://github. com/quantumlib/Cirq/graphs/contributors, doi: 10.5281/zenodo.6599601.
6. M. S. Anis et al., "Qiskit: An open-source framework for quantum computing," 2019, doi: 10.5281/zenodo.2562111. [Online]. Available: https://zenodo.org/record/2562111#.Y4fNnOzMLsI
7. Message Passing Interface Forum, MPI: A. Message-Passing Interface Standard Version 4.0, Jun. 2021. [Online]. Available: https://www.mpi-forum.org/docs/mpi-4.0/mpi40-report.pdf
8. OpenMP Architecture Review Board, "OpenMP application program interface version 5.2," Nov. 2021. [Online]. Available: https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-5-2.pdf
9. A. McCaskey, D. Lyakh, E. Dumitrescu, S. Powers, and T. Humble, "Xacc: A system-level software infrastructure for heterogeneous quantum-classical computing," *Quantum Sci. Technol.*, vol. 5, no. 2, 2020, Art. no. 024002, doi: 10.1088/2058-9565/ab6bf6.

MARTIN SCHULZ is a full professor of computer architecture and parallel systems in the Department of Computer Engineering, Technical University of Munich, 80333, Munich, Germany, and is on the Board of Directors at the Leibniz Supercomputing Centre, Bavarian Academy of Sciences and Humanities, 85748, Garching, Germany. Contact him at https://www.ce.cit.tum.de/caps/mitarbeiter/martin-schulz/ or schulzm@cit.tum.de.

MARTIN RUEFENACHT is a researcher for HPCQC integration in the Department of Quantum Computing and Technologies, Leibniz Supercomputing Centre, Bavarian Academy of Sciences and Humanities, 85748, Garching, Germany. Contact him at Martin.Ruefenacht@lrz.de.

DIETER KRANZLMÜLLER is the chairman of the Board of Directors, Leibniz Supercomputing Centre, Bavarian Academy of Sciences and Humanities, 85748, Garching, Germany, and a full professor for Communication Systems and System Programming, Institute of Informatics, Ludwig-Maximilians-Universität München, 80538, Munich, Germany. Contact him at https://www.mnm-team.org/~kranzlm/ or dieter.kranzlmueller@lrz.de.

LAURA SCHULZ is the head of strategic development and partnerships as well as the head of the Department of Quantum Computing and Technologies, Leibniz Supercomputing Centre, Bavarian Academy of Sciences and Humanities, 85748, Garching, Germany. Contact her at schulz@lrz.de.

IEEE COMPUTER SOCIETY

www.computer.org

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 society 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 society holds more than 215 conferences each year and sponsors many educational activities, including computing science accreditation.

Certifications: The society 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

IEEE CS EXECUTIVE COMMITTEE

President: Jyotika Athavale
President-Elect: Hironori Washizaki
Past President: Nita Patel
First VP: Grace A. Lewis
Second VP: Nils Aschenbruck
Secretary: Mrinal Karvir
Treasurer: Darren Galpin
VP, Member & Geographic Activities: Kwabena Boateng
VP, Professional & Educational Activities: Cyril Onwubiko
VP, Publications: Jaideep Vaidya
VP, Standards Activities: Edward Au
VP, Technical & Conference Activities: Terry Benzel
2023–2024 IEEE Division VIII Director: Leila De Floriani
2024–2025 IEEE Division V Director: Christina M. Schober
2024 IEEE Division V Director-Elect: Thomas M. Conte

IEEE CS BOARD OF GOVERNORS

Term Expiring 2024:
Saurabh Bagchi, Charles (Chuck) Hansen, Carlos E. Jimenez-Gomez, Daniel S. Katz, Shixia Liu, Cyril Onwubiko

Term Expiring 2025:
İlkay Altintaş, Mike Hinchey, Joaquim Jorge, Rick Kazman, Carolyn McGregor, Andrew Seely

Term Expiring 2026:
Megha Ben, Terry Benzel, Mrinal Karvir, Andreas Reinhardt, Deborah Silver, Yoshiko Yasuda

IEEE EXECUTIVE STAFF

Executive Director and COO: Sophia Muirhead
General Counsel and Chief Compliance Officer: Anta Cisse-Green
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: Kelly Lorne
Managing Director, Educational Activities: Jamie Moesch
IEEE Standards Association Managing Director: Alpesh Shah
Chief Financial Officer: Thomas Siegert
Chief Information Digital Officer: Jeff Strohschein
Managing Director, Conferences, Events, and Experiences: Marie Hunter
Interim Managing Director, Technical Activities: Ken Gilbert

IEEE OFFICERS

President & CEO: Thomas M. Coughlin
President-Elect: Kathleen Kramer
Past President: Saifur Rahman
Director & Secretary: Forrest D. Wright
Director & Treasurer: Gerardo Barbosa
Director & VP, Publication Services & Products: Sergio Benedetto
Director & VP, Educational Activities: Rabab Kreidieh Ward
Director & VP, Membership and Geographic Activities: Deepak Mathur
Director & President, Standards Association: James E. Matthews III
Director & VP, Technical Activities: Manfred J. Schindler
Director & President, IEEE-USA: Keith A. Moore

IEEE

revised 29 May 2024

EDITOR: Tim Menzies, North Carolina State University, tim@menzies.us

This article originally appeared in IEEE Software vol. 40, no. 1, 2023

DEPARTMENT: SE FOR AI

Low Code for Smart Software Development

Jordi Cabot and Robert Clarisó

FROM THE EDITOR

The more we know about patterns in code, the better we can support those patterns. In this article, Jordi Cabot and Robert Clarisó discuss the promise and perils of low-code environments that allow programmers and nonprogrammers alike to quickly deliver artificial intelligence (AI)-enhanced software solutions. They offer a "wish list" that outlines what developers need to watch for in low-code tools for smart software.

Got anything else you want to say about software engineering (SE) plus AI? For SE-plus-AI applications, do you have a surprising result or industrial experience, something that challenges decades of conventional thinking in SE? If so, e-mail a one-paragraph synopsis to tim@menzies.us [use the subject line "SE for AI: Idea: (Your Idea)"]. If that looks interesting, I'll ask you to submit a 1,000–2,400-word article (where each graph, table, and figure is worth 250 words) for review for *IEEE Software*. Note: heresies are more than welcome (if supported by well-reasoned industrial experiences, case studies, and other empirical results).—*Tim Menzies*

To tame complexity for many applications, including artificial intelligence (AI)-based systems, software engineers typically choose to work at a higher abstraction level,[1] where irrelevant technical details can be ignored, at least during the initial development phases. Low-code platforms are the latest incarnation of this trend, promising to accelerate software delivery by dramatically reducing the amount of hand coding required.

Low code can be seen as a continuation of other model-based approaches.[2] This includes the well-known model-driven architecture by the Object Management Group (https://www.omg.org/mda/). The benefits of low code are varied, ranging from faster prototyping and development to improved understanding, reusability, and maintenance.[3,4,5] Low-code platforms are especially promising in the current software landscape, where many software systems embed AI components, mostly based on ML techniques. These benefits are widely recognized, so much that low-code platforms are the now the basis for start-ups reaching billion-dollar valuations (see examples in the following).

Such AI-enhanced ML-enabled systems (also called *smart software*) give rise to unique software engineering challenges;[6,7,8] e.g., AI elements are hard to specify,[9] architect, test, and verify.[10] Organizations must adapt to leverage them.[11,12] Additional complexity arises from all the potential interactions between the AI components and the "traditional" ones (since we need to specify how they collaborate, test that they behave consistently, and analyze their interdependencies). Accordingly, in this article, we offer a "wish list" that outlines what developers need to watch for in low-code tools for smart software. Also, we present work on an architecture (see Figure 1) that is one way to satisfy items on that wish list.

STATE OF THE ART

Organizations trying to offer AI products and services are constrained by the challenges in attracting skilled

Digital Object Identifier 10.1109/MS.2022.3211352
Date of current version: 23 December 2022

 Published by the IEEE Computer Society

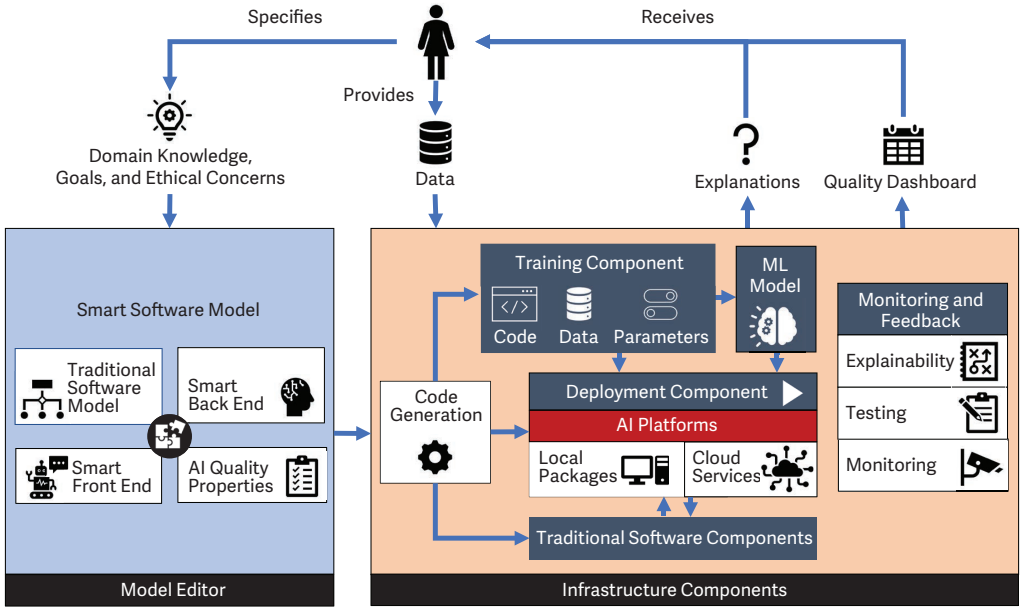


FIGURE 1. The low-code architecture discussed in this article.

talent. In a recent survey,[13] 83% of companies identified AI as a strategic priority but had problems enrolling talent. Another survey[14] reported that 40% of companies claim that AI technologies and expertise are too expensive. For instance, the U.S. Bureau of Labor Statistics[15] reported that the current mean average wage for data scientists in the United States was US$108,660. As a result, companies are looking into tools and technologies to 1) streamline the creation and management of AI products and services and 2) enable contributions from professionals without a strong AI background.

Low-code tools could be the solution to this problem. The number of low-code tools keeps growing.[16] Well-known examples include Mendix, OutSystems, Appian, and GeneXus, all with a significant user base (for example, OutSystems, Appian, and Mendix have all disclosed surpassing $100 million in annual recurring revenue), recent acquisitions (Mendix has been acquired by Siemens and GeneXus by Globant), and funding rounds (OutSystems raised more than half a billion euros in the past five years, with a US$9.5 billion valuation in the last round). Moreover, all major tech companies, including the "Big Five," have their own offering in this space, mostly oriented to help clients use their own tech stack and services when building new applications. Some vendors focus on vertical domains (e.g., human resources and customer relationship management automation) but most offer a generic solution for the horizontal domain of web-based and mobile data-intensive applications.

But, perhaps surprisingly, low-code tools have, so far, paid little attention to the new breed of smart software systems. Support for AI features is mostly constrained to embed and call an external AI service from one of the web components or back-end services. Such AI services are not modeled as part of the application. Instead, they are developed and deployed outside the "main" software application and integrated as black-box components. To minimize the amount of glue code required for this integration, some tools offer predefined integrations with specific providers. For instance, for a low-code tool to be able to work with Google's Dialogflow, creating a chatbot for a web application would require creating and deploying the chatbot logic with Dialogflow and using the low-code tool connector to provide the Dialogflow agent with credentials to display the bot on the desired webpage. Similarly, you could call other types of AI components and pretrained ML models, e.g., for computer vision, text classification, and sentiment analysis.

While, with some manual effort, developers can make it work, this solution is far from ideal. The software specification and code are scattered, and there is no visibility (or traceability and explainability) of

what happens inside the AI components, which have their own independent monitoring and analysis tools. Moreover, we are unable to analyze and describe the fine-grained details of the important collaboration aspects between the two and the potential side effects of such interactions. For instance, consider a loan approval component in a banking application that depends on an ML-based prediction system to decide which clients are to be trusted. A new release of the prediction model can drastically change the behavior of the application, but if that ML model is a black box, it is very difficult to account and test for this type of evolution.

CLEARLY, AI FEATURES ARE NOT FIRST-CLASS CITIZENS IN CURRENT LOW-CODE PLATFORMS: WE CANNOT MODEL HOW THE ML MODELS ARE TRAINED AND CONFIGURED.

Clearly, AI features are not first-class citizens in current low-code platforms: we cannot model how the ML models are trained and configured. We cannot decide what concrete low-level AI platform we want to target during deployment. And the list goes on. Smart software development requires low-code tools with a more native support for the specification of these systems. We have seen this type of abstraction capability in tools targeting only data science processes, such as the Konstanz Information Miner, Amazon SageMaker Studio, IBM SPSS Modeler, and Azure Machine Learning Studio. Therefore, we know it is possible. We next elaborate on this wish list for our ideal low code for smart software solutions.

LOW CODE FOR SMART SOFTWARE WISH LIST

We believe a developer working on smart software would be interested in a low-code platform capable of

- managing concerns for both AI components and traditional software components in a consistent and integrated way, including their interdependencies (e.g., an AI component trained using the data entered via a regular component)
- supporting the complete life cycle of the required AI components (training, validation, deployment, and monitoring) as well as tracing the decisions behind their architecture and evolution
- operating with a technology-independent and platform-agnostic specification while supporting a transparent deployment to different AI service providers
- enabling the integration of AI components in both the front end (e.g., chatbots) and back end (e.g., prediction tasks) of a system
- defining high-level goals and quality concerns (e.g., fairness) that can be automatically tested and/or monitored after deployment
- facilitating use without intricate knowledge of the underlying AI techniques, offering mechanisms to automatically select a suitable method and (hyper)parameters for a particular usage scenario
- supporting a variety of AI tasks beyond text and image classification.

A LOW-CODE ARCHITECTURE FOR SMART SOFTWARE

To provide the features identified in the preceding wish list, we envision an architecture (see Figure 1) based around the following components:

- *Model editor*: Developers provide a description of the software system using a unified notation—a smart software model—which includes both traditional and smart elements:
  - a description of the application domain and the architecture of the software system, i.e., components and the relationships among them
  - a high-level description of the tasks to be performed by the AI components (classification, synthesis, and so on), target quality metrics (e.g., a desired precision/recall value), and concerns (e.g., ethical issues, resource budgets for training, and deployment) that should be considered
  - a description of the input data sources storing the domain knowledge, with emphasis on the identification of information relevant from the point of view of fairness (e.g., gender, religion, country of birth, and so on); moreover, the developer should be able to select predefined

policies for data preparation and cleaning (e.g., how to deal with null values).

- *Code generator*: The information provided in the model drives the generation of code implementing the different processes within the low-code tool. The developer may either select a particular technology or service provider or delegate this choice to the tool, which can leverage model annotations describing nonfunctional qualities, such as price and scalability, to make simple tradeoff decisions.
- *Training*: The code generator emits code to train an ML model, preparing training and validation datasets from the input data sources according to the resource budget. After training, the target quality metrics are measured, and any ethical constraint is checked.
- *Deployment*: The trained model is deployed on a particular AI platform, which can be either a cloud service from a variety of providers or a local AI package.
- *Traditional software components*: Software modules that do not integrate AI features are generated in the usual way. These modules interact with AI components through a dedicated application programming interface that encapsulates the specific deployment strategy.
- *Monitoring and feedback*: Finally, the AI components should be continuously monitored and tested after deployment to provide continuous feedback to the developer. This feedback should include
  - a dashboard displaying performance information about both the target quality metrics and resource usage
  - explanations regarding the decisions made by the AI components, linking back to the input data sources or tracing back to requirements in the input model.

All components in this architecture are feasible and already partially exist as separate elements in other low-code, AI, and monitoring platforms. Bringing them all together in a unified framework could be a force multiplier and a significant next step in lowering the barrier to entry for the next generation of smart software developers.

ALL COMPONENTS IN THIS ARCHITECTURE ARE FEASIBLE AND ALREADY PARTIALLY EXIST AS SEPARATE ELEMENTS IN OTHER LOW-CODE, AI, AND MONITORING PLATFORMS.

REFERENCES

1. G. Booch, "The history of software engineering," *IEEE Softw.*, vol. 35, no. 5, pp. 108–114, Sep./Oct. 2018, doi: 10.1109/MS.2018.3571234.
2. J. Cabot, "Positioning of the low-code movement within the field of model-driven engineering," in *Proc. Companion MODELS'20*, ACM, 2020, pp. 76:1–76:3, doi: 10.1145/3417990.3420210.
3. C. Verbruggen and M. Snoeck, "Practitioners' experiences with model-driven engineering: A meta-review," *Softw. Syst. Model.*, early access, Jun. 2022, doi: 10.1007/s10270-022-01020-1.
4. J. Sijtstra, "Quantifying the effectiveness of low-code development platforms in the Dutch public sector," M.S. thesis, LIACS, Leiden Univ., Leiden, The Netherlands, 2022.
5. J. Whittle, J. E. Hutchinson, and M. Rouncefield, "The state of practice in model-driven engineering," *IEEE Softw.*, vol. 31, no. 3, pp. 79–85, May/Jun. 2014, doi: 10.1109/MS.2013.65.
6. J. Bosch, H. H. Olsson, and I. Crnkovic, "Engineering AI systems: A research agenda," in *Artificial Intelligence Paradigms for Smart Cyber-Physical Systems*, A. K. Luhach and A. Elçi, Eds. Hershey, PA, USA: IGI Global, Jan. 2021, pp. 1–19, doi: 10.4018/978-1-7998-5101-1.ch001.
7. I. Ozkaya, "What is really different in engineering AI-enabled systems?" *IEEE Softw.*, vol. 37, no. 4, pp. 3–6, Jul./Aug. 2020, doi: 10.1109/MS.2020.2993662.
8. H. Muccini and K. Vaidhyanathan, "Software architecture for ML-based systems: What exists and what lies ahead," in *Proc. 2021 IEEE/ACM 1st Workshop AI Eng.- Softw. Eng. AI (WAIN)*, pp. 121–128, doi: 10.1109/WAIN52551.2021.00026.
9. M. Rahimi, J. L. C. Guo, S. Kokaly, and M. Chechik, "Toward requirements specification for machine-learned components," in *Proc. 2019 IEEE 27th Int. Requirements Eng. Conf. Workshops (REW)*, pp. 241–244, doi: 10.1109/REW.2019.00049.
10. V. Riccio, G. Jahangirova, A. Stocco, N. Humbatova, M. Weiss, and P. Tonella, "Testing machine learning based systems: A systematic mapping," *Empirical Softw. Eng.*,

vol. 25, no. 6, pp. 5193–5254, Nov. 2020, doi: 10.1007/s10664-020-09881-0.
11. D. Dahlberg, "Developer experience of a low-code platform: An exploratory study," M.S. thesis, Umeå universitet, Samhällsvetenskapliga fakulteten, Institutionen för Informatik, Umeå, Sweden, 2022.
12. A. Vallecillo, On the industrial adoption of model driven engineering. Is your company ready for MDE? *Int. J. Inf. Syst. Softw. Eng. Big Companies*, vol. 1, no. 1, pp. 52–68, Aug. 2015.
13. S. Ransbotham, D. Kiron, P. Gerbert, and M. Reeves, "Reshaping business with artificial intelligence: Closing the gap between ambition and action," *MIT Sloan Manage. Rev.*, vol. 59, no. 1, pp. 1–22, Sep. 2017.
14. "State of cognitive survey," Deloitte, Aug. 2017. [Online] Available: https://www2.deloitte.com/us/en/pages/deloitte-analytics/articles/cognitive-technology-adoption-survey.html
15. "Occupational employment and wages, May 2021—15-2051: Data scientists," U.S. Bureau of Labor Statistics, Washington, DC, USA, 2021. Accessed: Oct. 3, 2022. [Online.] Available: https://www.bls.gov/oes/current/oes152051.htm
16. A. Bucaioni, A. Cicchetti, and F. Ciccozzi, "Modelling in low-code development: A multi-vocal systematic review," *Softw. Syst. Model.*, vol. 21, no. 5, pp. 1–23, Oct. 2022, doi: 10.1007/s10270-021-00964-0.

JORDI CABOT is an ICREA Research Professor at the Internet Interdisciplinary Institute, Universitat Oberta de Catalunya, Barcelona 08018, Spain, where he leads the Software and Systems Modeling Lab. Contact him at https://jordicabot.com or jordi.cabot@icrea.cat.

ROBERT CLARISÓ is an associate professor with the Faculty of Computer Science, Multimedia, and Telecommunications, Universitat Oberta de Catalunya, Barcelona 08018, Spain, where he is also a member of the SOM Research Lab at the Internet Interdisciplinary Institute. Contact him at https://robertclariso.github.io or rclariso@uoc.edu.

IEEE COMPUTER SOCIETY D&I FUND

Drive Diversity & Inclusion in Computing

Supporting projects and programs that positively impact diversity, equity, and inclusion throughout the computing community.

DONATE TODAY!

IEEE COMPUTER SOCIETY

IEEE Foundation

EDITORS: Anshu Dubey, adubey@anl.gov
Konrad Hinsen, konrad.hinsen@cnrs.fr

DEPARTMENT: SCIENTIFIC PROGRAMMING

This article originally appeared in
Computing in Science & Engineering
vol. 24, no. 4, 2022

On the Role of Computer Languages in Scientific Computing

Dorian Leroy, *CEA, DAM, DIF, F-91297, Arpajon, France*
June Sallou, Johann Bourcier, and Benoit Combemale, *Université de Rennes, F-35000, Rennes, France*

Scientific codes are complex software systems. Their engineering involves various stakeholders using various computer languages for defining artifacts at different abstraction levels and for different purposes. In this article, we review the overall processes leading to the development of scientific software, and discuss the role of computer languages in the definition of the different artifacts. We provide guidelines to make informed decisions when the time comes to choose a computer language to develop scientific software.

Scientific computing is a cross-cutting field, its heart and soul being the development of mathematical models to understand physical systems through their simulations. Those models can be numerical (e.g., systems of differential equations), nonnumerical (e.g., agent-based models) or based on analytics (e.g., machine learning models), and aim to capture the behavior of the modeled system. Numerical models can be further refined as continuous or discrete. Simulations of mathematical models correspond to the execution of the computer programs containing these models, the so-called "simulation codes." In this article, we refer to the subsuming concept of *scientific software*, which we define as software dedicated to scientific computing and simulation. The development of scientific software involves both software engineering (SE) and scientific computing concerns. Mathematical models and scientific software are tightly coupled throughout their life cycles. The tools and methods used for their development—in particular, computer languages—can impact the definition of both, as well as the engineering principles that ensure the development of reliable scientific software.

When the time comes to implement a new model, i.e., develop new simulation software, scientists and engineers are faced with the choice of what computer language(s) to use (e.g., MATLAB, Mathematica, Fortran, Python, C++, or even an in-house domain-specific language). This choice has important consequences on the expressiveness available to implement the model and the corresponding simulation code, but also in terms of SE practices to develop reliable and efficient scientific software. The more general-purpose the language is—with low-level, computing-related, system abstractions—the more flexibility and performance it may provide, but also the more rigorous engineering principles and verification & validation (V&V) activities will be required to obtain a reliable piece of scientific software.

Most scientists and engineers are not trained in SE and are therefore not aware of its best practices beyond programming (e.g., version control management, component reuse, unit testing, continuous integration),[1] which has led to initiatives addressing this problem, such as the research software engineering movement.[2] Since the final goal is to build and apply the model encoded in the simulation code, the code itself is merely a means to that end. Final stakeholders (e.g., citizens, policy, and decision makers, research institutions, system users) may even be unaware of the importance of software for science and engineering.

In this article, we explore the overall scientific software development process, we provide an integrated view of the scientific computing and SE activities, artifacts, and roles, and we discuss the tradeoffs on the computer languages at hand to help scientists and engineers make informed decisions.

1521-9615 © 2023 IEEE
Digital Object Identifier 10.1109/MCSE.2022.3221672
Date of current version 4 January 2023.

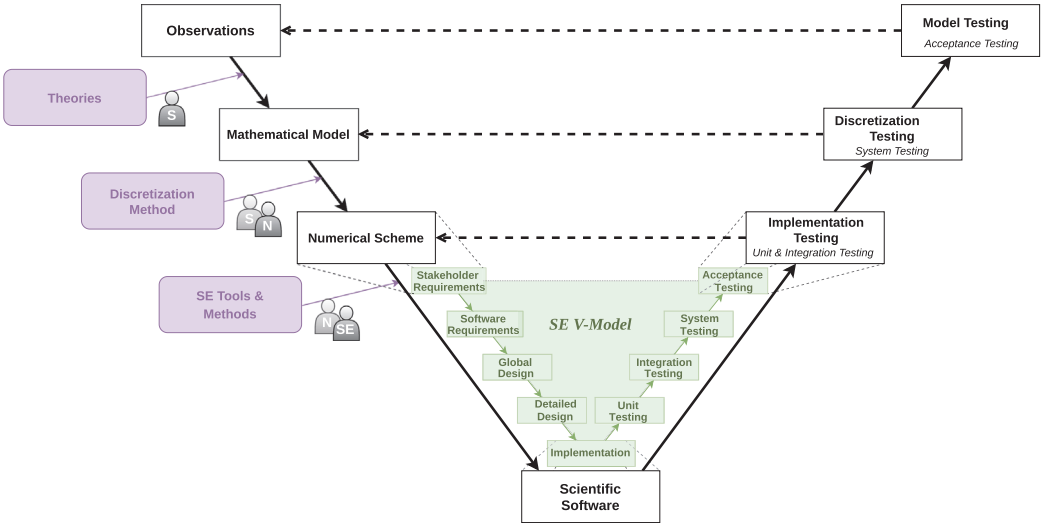


FIGURE 1. Overall scientific software development process across the scientific V-model.[4]

SCIENTIFIC COMPUTING: COMPUTER LANGUAGES TO THE RESCUE

The implementation of scientific software is the result of the successive refinement of different artifacts, starting with *observations* to elaborate the *mathematical model* thanks to *theories*, then, applying *discretization methods* to obtain a *numerical scheme*, to finally end with the implementation of the *scientific software* (cf. Figure 1).

Thus, the design of scientific software based on mathematical models requires the involvement and cooperation of various stakeholders, ranging from scientists and engineers to experts in numerical analysis or SE. These stakeholders play one of three roles (depending on context, one person may fulfill more than one role): *scientists* as domain experts, *numerical analysts* as experts on the discretization of a continuous phenomenon, and *software engineers* as experts on software development to deliver the expected services. Each role is in charge of the elaboration of one of the artifacts: scientists define the mathematical model, numerical analysts define the numerical scheme, and software engineers implement the computer software.

Computer languages enable the different stakeholders to perform their activities at the corresponding level of abstraction. We can thus classify computer languages according to their level of abstraction and the support they provide to stakeholders.

Languages to Define the Mathematical Model

Scientists can define a *mathematical model* and derive the corresponding *scientific software* using languages such as Mathematica,[a] or MATLAB.[b] Such languages provide continuous mathematical constructs (e.g., algebraic computation and differential blocks in MATLAB's block diagrams) allowing scientists to directly define their mathematical models with the language. The language infrastructure is then able to automatically discretize the mathematical models defined with the language, possibly in a configurable way, and to derive the corresponding scientific software.

Languages to Specify the Numerical Scheme

Alternatively, some languages allow deriving scientific software directly from a numerical scheme. Languages dedicated to the definition of *numerical schemes* (or with the right abstractions to do so), such as Julia,[c] R,[d] or NabLab,[e] allow automatically deriving the corresponding piece of scientific software without having to handle SE concerns. Thus, once numerical analysts obtain a numerical scheme as a result of the application of their chosen discretization method to the mathematical model, they can directly implement it using the discrete mathematics constructs offered by the language. From this encoded numerical scheme, the infrastructure of the language (e.g., model transformations, interpreters, compilers, code generators) derives the corresponding piece of scientific software.

[a]https://www.wolfram.com/mathematica or more specifically the Wolfram language, https://www.wolfram.com/language
[b]https://matlab.mathworks.com
[c]https://julialang.org/
[d]https://www.r-project.org/
[e]https://cea-hpc.github.io/NabLab/

Languages to Implement the Scientific Software

Finally, when software engineers deal with execution-related concerns (e.g., architecture, hardware, optimization, storage, etc.), system-level languages such as C,[f] C++,[g] and Fortran[h] can be used, together with frameworks like OpenMP,[i] and standards such as MPI.[3] Language users express the particularities of their simulator with regard to all the concerns involved in the development of scientific software, ranging from the mathematical model to the encoded numerical scheme and system-level concerns such as concurrency, memory, and data handling.

The artifacts at each level of abstraction can capture different concerns (e.g., data curation, mesh definition, and numerical analysis all relate to a numerical scheme, while concurrency and memory management are both related to the scientific software). Capturing these different concerns in a given artifact can be achieved with a single *general-purpose* language, or with separate, though coordinated, *dedicated* languages, leading to a polyglot development of this artifact.

The successive refinement of the different artifacts can be done automatically by the language infrastructure provided by interpreters or compilers. It can also be done (at least partially) manually by the different stakeholders, by specifying the handling of certain concerns according to their specific expertise. While automatic refinement through the language infrastructure provides a predefined way of refining a given artifact, manual refinement lets different roles handle concerns on their own and optimize their implementation for a given context.[4] For instance, a numerical scheme specified with NabLab is usually compiled using one of the compilation chains, thereby automatically taking into account the execution flow, parallelism, and memory model. Yet, one may want to handcraft the C++ code generated from a NabLab specification to customize how the related concerns are handled in a particular application.

COMPUTER LANGUAGES: V&V TECHNIQUES TO THE RESCUE

Language choice allows selecting the level of abstraction at which one wants to work. This determines which artifacts must be defined as part of the development process, and which artifacts are automatically derived through the language infrastructure. While this language infrastructure guarantees the correctness of the derived artifacts with regard to user-defined ones, the V&V concerns corresponding to those user-defined artifacts still need to be addressed.

For example, using a language at the discrete-mathematics abstraction level, numerical analysts can derive the scientific software from the numerical scheme. This derived software is guaranteed to be correct with regard to the provided numerical scheme, but the correctness of both the numerical scheme and the governing equations constituting the mathematical model still remains to be assessed.

We illustrate this on Figure 1, a V-Model for scientific computing, or scientific V-Model, where the different artifacts involved in scientific software development are represented on the left, from observations to mathematical model, numerical scheme, and actual scientific software. Facing each of these artifacts is the corresponding V&V concern to be addressed. In addition, for each artifact and V&V concern, the figure indicates the associated roles, i.e., the skills necessary to develop the artifacts and address their corresponding V&V concerns.

The model contains a nested V-model ("SE V-model") representing the artifacts specific to SE that are defined over the course of the development of the actual scientific software, from stakeholder requirements to the implementation. This nested SE V-model also contains the SE-specific V&V activities required to address the V&V concerns corresponding to each of these SE-specific artifacts.

The scientific V-model reads as follows. The left descending branch of the V-Model indicates which artifacts must be defined, based on the level of abstraction at which one works: artifacts above the chosen level of abstraction have to be defined as well, as each acts as specification for the artifact directly below. The right ascending branch of the V-model indicates the V&V activities to be undertaken for each artifact defined by the stakeholders. This includes the V&V activities corresponding to the artifacts defined with the chosen language, and every V&V activity situated above. In addition, while languages provide guarantees over the software they allow deriving, any V&V activity not handled by a language is left to the developers.

For a more detailed look at the scientific V-model, we direct the reader to our previous work.[4]

CLASSIFYING COMPUTER LANGUAGES FOR SCIENTIFIC COMPUTING

In Table 1, we propose a guide supporting decision-making with regard to the computer language(s) to

[f] https://www.iso.org/standard/74528.html
[g] https://isocpp.org/
[h] https://fortran-lang.org/
[i] https://www.openmp.org/

TABLE 1. Overview of languages commonly used in scientific computing according to the associated levels of abstraction.

Language	Mathematical Model	Numerical Scheme	Scientific Software
Mathematica (Wolfram Language)	+++	++	
MATLAB	++	++	
R	+	+	
NabLab		+++	
Julia		++	+
SciPy		++	+
Python			+
Java			++
C/C++			+++
Fortran		++	+++

use for scientific software developments. We evaluate a range of computer languages commonly used in scientific computing,[5,6] aiming to highlight how well each language supports the definition of the three categories of artifacts (mathematical model, numerical scheme, and scientific software), and how they facilitate the required V&V activities for these artifacts.

We propose a scale assigning a score to each language for the development of the three identified artifacts based on their ability to accurately describe these artifacts and the level of expertise required for their use. The more "+" symbols, the more detailed the language can describe the corresponding artifact, and the more expertise it requires from the designer. For example, languages providing fine control over concurrency and memory are better suited if one needs to directly work at the system level to define the scientific software (e.g., for performance or architecture reasons). In the remainder of this section, we give a brief overview of these languages.

The Wolfram Language, provided as part of Mathematica, offers continuous mathematical constructs, while also providing some expressivity with regard to discrete mathematics.[j] MATLAB works similarly, but also provides discrete numerical constructs.[k] R is a language more geared toward statistics, but can also be used for matrix computations, and provides continuous mathematical abstractions as well.[l] For each of these languages, SE abstractions are mostly kept out of the hands of the language user, and the associated concerns are addressed as part of their supporting infrastructure.

NabLab is a language dedicated to numerical analysis, which provides code generators targeting an array of C++ backends.[7] The language exclusively exposes numerical abstractions, and SE concerns are addressed as part of the provided generators and compilation chains.

Julia is a language gaining traction in scientific computing. It offers numerical abstractions, while giving finer control over some system concerns as well, such as multithreading and networking, enabling its use in the context of high-performance computing.[m] However, when such system-level abstractions are used, the user needs to address the corresponding V&V concerns as usual, requiring SE skills.

Python is a popular language in scientific computing, even if it does not provide native abstractions suited to continuous or discrete mathematics. This popularity stems from its low entry level in terms of SE skills (e.g., dynamic typing, managed memory), its extensive library support, such as SciPy[n] and NumPy[o] providing the missing abstractions for scientific computing, and a mature support for the definition of wrappers for C/C++ applications.

Java does not natively provide mathematical abstractions, but abstracts some system-level concerns, such as memory management. It is also cited in the literature as one of the frequently used languages by the scientific community.[6]

C and C++ are extensively used in the scientific computing community, despite missing numerical and mathematical abstractions, and working at a very low level of abstraction.[8] This is due to its good performance and the large number of libraries available for scientific computing. However, developing scientific software with C or C++ demands addressing numerous SE V&V concerns, which come in addition to the usual numerical and mathematical V&V concerns. Fortran is a similar case to C and C++, except that it does provide numerical abstractions, as it was designed for writing scientific software.[p]

CONCLUSION

In this article, we presented a scientific software development process that integrates both scientific computing and SE activities: the scientific V-model. This model describes how the different artifacts and stakeholders involved in this process are related to

[j]https://www.wolfram.com/language/index.php.en
[k]https://www.mathworks.com/products/matlab.html
[l]https://www.r-project.org/

[m]https://julialang.org/
[n]https://scipy.org/
[o]https://numpy.org/
[p]https://fortran-lang.org/

each other. We also provide a categorization of computer languages according to their ability to develop specific artifacts and discuss the impact on verification and validation activities.

We argue that, when choosing a computer language to implement scientific software, the researcher should consider the level of abstraction at which they are working and keep in mind that this choice has an impact on the V&V activities they must manage. To facilitate an informed decision on the choice of computer languages, we provide a guide with the most commonly used languages in scientific computing, referring to the skills required to take full advantage of them in the definition of artifacts and associated V&V activities.

With this article, we aim to make scientific computing practitioners aware of the role of computer languages and to initiate the discussion early on with the information necessary when faced with the choice of computer language(s) to use in scientific computing.

ACKNOWLEDGMENTS

The author would like to thank their colleagues at the Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA), Geoscience Department of the Observatoire des Sciences de l'Univers de Rennes (OSUR), as well as at the Commissariat à l'énergie atomique et aux énergies alternatives (CEA), for their input and providing such interesting conversations that contributed to the writing and improvement of this article.

REFERENCES

1. G. Wilson et al., "Best practices for scientific computing," *PLoS Biol.*, vol. 12, no. 1, 2014, Art. no. e1001745, doi: 10.1371/journal.pbio.1001745.
2. J. Cohen, D. S. Katz, M. Barker, N. C. Hong, R. Haines, and C. Jay, "The four pillars of research software engineering," *IEEE Softw.*, vol. 38, no. 1, pp. 97–105, Jan./Feb. 2021, doi: 10.1109/MS.2020.2973362.
3. W. Gropp, E. Lusk, N. Doss, and A. Skjellum, "A high-performance, portable implementation of the MPI message passing interface standard," *Parallel Comput.*, vol. 22, no. 6, pp. 789–828, 1996, doi: 10.1016/0167-8191(96)00024-5.
4. D. Leroy, J. Sallou, J. Bourcier, and B. Combemale, "When scientific software meets software engineering," *Computer*, vol. 54, no. 12, pp. 60–71, 2021, doi: 10.1109/MC.2021.3102299.
5. P. Prabhu et al., "A survey of the practice of computational science," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, 2011, pp. 1–12, doi: 10.1145/2063348.2063374.
6. E.-M. Arvanitou, A. Ampatzoglou, A. Chatzigeorgiou, and J. C. Carver, "Software engineering practices for scientific software development: A systematic mapping study," *J. Syst. Softw.*, vol. 172, 2021, Art. no. 110848, doi: 10.1016/j.jss.2020.110848.
7. B. Lelandais, M. P. Oudot, and B. Combemale, "Fostering metamodels and grammars within a dedicated environment for HPC: The NabLab environment (tool demo)," in *Proc. 11th ACM SIGPLAN Int. Conf. Softw. Lang. Eng.*, 2018, pp. 200–204, doi: 10.1145/3276604.3276620.
8. J. Pitt-Francis and J. Whiteley, *Guide to Scientific Computing in C++*. New York, NY, USA: Springer, 2017, doi: 10.1007/978-3-319-73132-2.

DORIAN LEROY is a research engineer at CEA, F-91297, Arpajon, France. His research interests include field of Software Language Engineering and include metaprogramming approaches and V&V facilities. Contact him at dorian.leroy@cea.fr.

JUNE SALLOU is a postdoctoral researcher in software engineering at University of Rennes 1, F-35000, Rennes, France. Her research interests include scientific modeling, approximate computing, and environmental science. Contact her at june.benvegnu-sallou@univ-rennes1.fr.

JOHANN BOURCIER is an associate professor of software engineering at University of Rennes 1, F-35000, Rennes, France. His research interests include software engineering include self-adaptive systems, model-driven engineering, and distributed and heterogeneous software environments. Contact him at johann.bourcier@irisa.fr.

BENOIT COMBEMALE is a full professor of software engineering at University of Rennes 1, F-35000, Rennes, France. His research interests include software engineering include software language engineering, model-driven engineering, and software validation and verification. Contact him at benoit.combemale@inria.fr.

IEEE Computer Society Has You Covered!

WORLD-CLASS CONFERENCES — Stay ahead of the curve by attending one of our 195+ globally recognized conferences.

DIGITAL LIBRARY — Easily access over 900k articles covering world-class peer-reviewed content in the IEEE Computer Society Digital Library.

CALLS FOR PAPERS — Discover opportunities to write and present your ground-breaking accomplishments.

EDUCATION — Strengthen your resume with the IEEE Computer Society Course Catalog and its range of offerings.

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

NETWORK — Make connections that count by participating in local Region, Section, and Chapter activities.

Explore all of the member benefits at www.computer.org today!

IEEE COMPUTER SOCIETY

IEEE

EDITOR: Nir Kshetri, University of North Carolina at Greensboro, nbkshetr@uncg.edu

DEPARTMENT: COMPUTING'S ECONOMICS

This article originally appeared in
Computer
vol. 55, no. 9, 2022

Democratizing Science Through Advanced Cyberinfrastructure

Manish Parashar, *National Science Foundation*

Democratizing access to cyberinfrastructure is essential to democratizing science. This article explores knowledge, technical, and social barriers to accessing and using cyberinfrastructure and explores approaches to addresses them. It also highlights recent activities and investments at the National Science Foundation that implement some of these approaches.

Science and engineering (S&E) research in the 21st century, powered by the growing availability of computation and data, continues to explore new frontiers, generating discoveries and innovations with the potential to transform our lives, our environment, and our economies. Examples of such S&E research enabled by computation and data include fundamental discoveries about our solar system and the universe, understanding and modeling climate change and potential strategies for mitigating these changes, understanding the nature and progression of diseases and how to cure illnesses, changing the way we farm and deliver food and other natural resources to consumers, and responding to and managing the impacts of natural disasters, such as hurricanes, earthquakes, and wildfires. Researchers today have unprecedented amounts of data from diverse sources, including sensors, instruments, and computational simulations, as well as an equally unprecedented need for computing to allow them to extract meaningful insights from the data to drive understanding, predictions, and decision making.

The essential role of computation and data in 21st century science has been highlighted in several recent reports, including a 2021 National Academies of Sciences, Engineering, and Medicine (NASEM) report, "Global Change Research and Opportunities for 2022–2031,"[1] which identified the critical need for computing research and computing resources that can advance the nation's understanding of and response to climate challenges, as well as the 2021 NASEM "Decadal Survey on Astronomy and Astrophysics,"[2] which summarized the tremendous impacts of data from multiple observatories. It is therefore imperative, now more than ever, that all researchers benefit from the opportunities for scientific exploration enabled by computation and data. As a result, ensuring broad, fair, and equitable access to advanced cyberinfrastructure (CI), including computing, data, networking, software, and expertise, that is, democratizing access to CI, is essential to democratizing science.

Recognizing this growing role of computation and data across all areas of scientific research, there have been significant investments around the globe in advanced CI resources, services, and expertise. In the United States, the National Science Foundation (NSF), through its Office of Advanced Cyberinfrastructure (OAC) and predecessor offices, has, over the past four decades, funded the development and provisioning of advanced CI resources and services toward an overarching vision[3] of ensuring the broad availability and innovative use of an agile, integrated, robust, trustworthy, and sustainable CI ecosystem that can drive new thinking and transformative discoveries in all areas of research and education. Investments include acquisition, integration, coordination, and production operations associated with shared data, secure networking, advanced computation, scientific software and data services, and the design and development of computational and data-enabled S&E.

Digital Object Identifier 10.1109/MC.2022.3174928
Date of current version: 29 August 2022

 Published by the IEEE Computer Society

The OAC also nurtures the computational and data skills and expertise needed for next-generation S&E research, and it promotes innovative, robust, secure, and interoperable CI as well as sharing and collaboration among academic research infrastructure groups, other federal agencies, international research funders, and the private sector. A recent road map by the National Science and Technology Council Subcommittee on Future Advanced Computing Ecosystems[4] highlights that these investments are complemented by investments at the national, regional, and local levels; by other U.S. federal agencies, academic institutions, and industry; and in other countries.

BARRIERS TO CI ACCESS AND USE

However, despite these global investments and the growing availability of an advanced CI ecosystem, significant barriers still limit broad and equitable access to this ecosystem, especially for individuals and institutions that are resource constrained and for communities that have been traditionally underrepresented. These knowledge, technical, and social barriers were explored in a recent study,[5] and they span several key areas, as summarized in the following.

KNOWLEDGE BARRIERS

As a result of increasing national, regional, and institutional investments in the CI ecosystem, researchers typically have access to a range of resources and services. However, researchers often lack a broad awareness of this availability, and perhaps even more importantly, even when they are aware of the available resources, they may not understand how to use these resources (for example, determining which resources are relevant to a researcher's specific needs and how to gain access to these resources can be challenging). Effective use of advanced computing and other CI resources requires significant support structures to help researchers determine the most appropriate resources, obtain allocations, train practitioners once they have allocations, and support application migration and execution. Such support structures are often missing at the local level, especially at underresourced institutions that often need the most help. The lack of access to necessary support has been noted as the most significant barrier to broad access to the CI ecosystem.[5] National and regional entities and institutions do provide support structures, but scaling these structures to meet the needs of a growing and diverse community is challenging. Current efforts often focus on domains and communities that have traditionally used advanced CI resources rather than on integrating new and developing communities that are not as able to adapt to specific needs.

BARRIERS STILL LIMIT BROAD AND EQUITABLE ACCESS TO THIS ECOSYSTEM, ESPECIALLY FOR INDIVIDUALS AND INSTITUTIONS THAT ARE RESOURCE CONSTRAINED AND FOR COMMUNITIES THAT HAVE BEEN TRADITIONALLY UNDERREPRESENTED.

A related barrier is associated with the recruitment, retention, and cultivation of a highly capable, adaptive, and agile workforce, for example, system administrators, software developers, and data curators. Developing and sustaining such a CI workforce presents challenges, including initiating effective institutional and on-the-job training to keep up with evolving software, technologies, platforms, and application requirements. Furthermore, there remains a lack of recognized job titles for the CI workforce, and skilled CI workers often face career uncertainty and a lack of recognition, as their valued services are in positions not visible to the research community and they have titles that are neither consistent nor meaningful.

Providing creative incentives, reward mechanisms, and career paths will be essential to sustain this workforce. Whereas the formal educational pathway is critical, providing on-ramps for nontraditional students (for instance, those who are older and/or seeking reskilling and upskilling opportunities) requires thoughtful educational practices, mentoring, and support to promote success and advancement.

TECHNICAL/PROCEDURAL BARRIERS

The number and diversity of researchers using CI resources has continued to evolve over the past few decades. However, the mechanisms used to allocate these resources have largely remained the same. Most national and regional resources and many institutional resources use proposal-based mechanisms for allocating resources: once their research is funded, researchers must submit a second proposal requesting resources. These proposals are then periodically evaluated, and if selected, the researchers are given an allocation that they can use to access the resources and conduct their research. This approach works well for certain classes of users and types of usage modes, but it can prevent broad usage of the resources for a multitude of reasons. For example, users often face "double jeopardy" by having to get their proposal and resource requests through two separate processes. Users must have a certain level of expertise and experience to appropriately articulate their needs and put together a competitive proposal. The latency of the proposal process presents additional barriers. Furthermore, proposal review criteria tend to be skewed toward more experienced users, requiring prior results for performance and scalability. Although many systems provide small "start-up" allocations, these are limited and cannot support extended research needs. Alternate access models, such as on-demand and urgent access as well as the integration of access into popular science tools, are not typically supported.

SPECIFIC EFFORTS AND INCENTIVES FOCUSED ON INCREASING AWARENESS AND ACCESS BY, FOR EXAMPLE, INTEGRATING AND EMBEDDING CI, CI EXPERTISE, AND CI BEST PRACTICES WITHIN COMMUNITIES, MUST BE A PRIORITY.

The ability of researchers to use growing national and regional CI capabilities is often limited by local infrastructure, which is typically needed to allow them to have access and effectively use these resources. These limits include a lack of adequate local capabilities for securely connecting to advanced computational resources, accessing relevant data resources, and integrating these resources into their application workflow. Perhaps most importantly, local resources are not equitably available across the full range of institution types, preventing certain segments of the research community from accessing the on-ramps that would pave the way toward their engagement in computational and data-enabled S&E.

Social barriers

In addition to the technical and procedural barriers noted in the preceding, there remain social barriers at the institutional and regional levels that impact how research CI is viewed, funded, and supported. For example, the unique nature of research CI, how it is used, the needs of its user community, and how it differs from more typical IT infrastructure and services are often not appreciated at an institutional level, resulting in a lack of mechanisms and structures needed to support researchers and, more importantly, to expose them to the potential benefits of CI to their research. This lack of appreciation also makes it harder to attract and retain the necessary talent and can lead to the deployment of CI solutions that do not match user needs. Furthermore, these adverse impacts often disproportionately affect underresourced institutions and communities. The resulting lack of engagement of underresourced institutions and communities can further result in the downstream exclusion of certain communities and their contributions from the scientific research enterprise and the propagation of bias. Specific efforts and incentives focused on increasing awareness and access by, for example, integrating and embedding CI, CI expertise, and CI best practices within communities, must be a priority.

DEMOCRATIZING ACCESS TO CI

Democratizing access to the CI ecosystem is essential to democratizing science and ensuring that every researcher has fair and equitable access to the resources that support his or her work. As the needs for and opportunities from CI grow and broaden, eliminating the barriers listed previously is becoming critical. This approach requires strategic investment in a broad set of CI resources, services, and expertise that can systematically address barriers to CI access while, at the same time, providing adequate training and support structures. Specifically, such investments should consist of the following:

- integrated and user-friendly portals and gateways for discovering and accessing resources, supported by flexible allocation and access mechanisms that sustain a wide spectrum of users and their needs
- access to local CI resources as part of a shared fabric of national CI resources connected and reachable through high-speed frictionless data networking
- diverse and flexible allocation and access modes (for example, on-demand, urgent, and coordinated access) that support a diversity of users and application needs
- agile, easily accessible, and scalable networks of experts that integrate embedded expertise and user support that is responsive to local needs
- broadly accessible training targeting the spectrum of CI users and skills as well as support for exchanges among communities and the dissemination of best practices.

Collectively, the preceding steps can be transformational in broadening and democratizing access to CI and the research opportunities CI provides.

NSF investments toward democratizing CI

The broad requirements for democratizing CI listed in this article are fundamental to the NSF's CI vision[3] for a national CI ecosystem and underlie many of the agency's recent investments that implement this vision. They are also integral to the NSF's blueprint for future national CI coordination services.[6] For example, the NSF recently announced[7] a suite of awards through its Advanced Cyberinfrastructure Coordination Ecosystem: Services and Support (ACCESS) program. This program is aimed at improving the accessibility and usability of the national CI ecosystem and increasing its integration with systems and research communities on campuses across the nation. ACCESS services build on the NSF's past CI investments and activities as well as more recent explorations, such as a high-throughput computing allocation pilot.[8]

DEMOCRATIZING ACCESS TO THE CI ECOSYSTEM IS ESSENTIAL TO DEMOCRATIZING SCIENCE AND ENSURING THAT EVERY RESEARCHER HAS FAIR AND EQUITABLE ACCESS TO THE RESOURCES THAT SUPPORT HIS OR HER WORK.

Complementing the ACCESS services are the NSF's sustained efforts to foster and nurture a diverse, recognized, and skilled CI professional (CIP) workforce.[9] *CIP* refers to the community of individuals who provide a broad spectrum of skills and expertise to the scientific and engineering research enterprise by inventing, developing, deploying, and/or supporting research CI and CI users. Examples of CIPs include CI system administrators, CI research support staff, CI research software engineers, data curators, and CI facilitators, and it may also include computational research scientists and engineers who are not in traditional academic paths. The NSF uses the broadest definition of CIP, including researchers who use CI, CI developers, and CI operators—all workforce categories required to effectively leverage and utilize current, emerging, and future CI capabilities and amplify the transformative impact of CI across S&E research fields.[10,11] These individuals and the highly valued services they provide to S&E deserve more institutional recognition, support as a community, and clearer pathways for their professional/career development. Specific NSF activities to support CIPs include the Training-Based Workforce Development for Advanced Cyberinfrastructure (CyberTraining) program,[12] which

supports innovative, scalable training, education, and curriculum/instructional materials along with deeper incorporation of CIPs into the research enterprise.

CyberTraining seeks to broaden CI access and adoption by 1) increasing adoption of advanced CI and computational and data-driven methods by a broader range of S&E disciplines and institutions; 2) enhancing the incorporation of CIPs into the research enterprise, highlighting the value of those professionals in S&E research; and 3) effectively utilizing the capabilities of individuals from a diverse set of underrepresented groups. The program includes a track for funding CIPs at the institute and regional levels and their integration into a national computational science support network managed by one of the services that are part of the ACCESS program. Importantly, CyberTraining and other programs supporting CIPs require a mentoring and/or professional development plan to encourage research proposals to explicitly consider and support this important but often neglected aspect of CI.

A related support activity is the Research Computing and Data Nexus Cyberinfrastructure Center of Excellence,[13,14] which aims to advance research computing and data infrastructure through the strategic development of tools, practices, and professionals. Overall, the NSF envisions networks of connected and coordinated hubs that recognize and connect CIPs, support communications and training, share best practices, and foster mobility and synergies across projects and organizations. Finally, the NSF recognizes the importance of diversity in driving scientific innovation and discovery. The NSF is thus committed to enabling the broadest access to its CI ecosystem and continues to make investments to support this commitment. For example, the recently funded Minority-Serving Cyberinfrastructure Consortium[15] envisions a transformational partnership to promote advanced CI capabilities on the campuses of historically Black colleges and universities, Hispanic-oriented institutions, tribal colleges and universities, and other minority serving institutions.

To harness the full potential of research discoveries and the resulting impacts on science and society, all researchers must be able to avail themselves of the opportunities for scientific exploration provided by advanced CI. Ensuring broad, fair, and equitable access to advanced CI resources, services, and expertise will be essential for democratizing science in the 21st century. This article described some of the barriers to achieving this objective on the basis of experiences at the NSF and highlighted the NSF's recent activities and investments to address these barriers. However, the democratization of science and its benefits requires not only continued investments by the NSF but also broader and more coordinated local, national, and global efforts and investments. The NSF intends to continue to look at more effective ways to reduce barriers to CI access and explore new approaches to ensure broader participation and equity. For example, the NSF is coleading the congressionally chartered National Artificial Intelligence (AI) Research Resource Task Force,[16] which is seeking to address the "resource divide" in AI research and has been developing a plan to democratize access to AI R&D for America's researchers and students and providing them with critical computational, data, and training resources through a broadly accessible shared CI.

ACKNOWLEDGMENT

The author is grateful for discussions with and thoughtful feedback from NSF colleagues about drafts of this article.

REFERENCES

1. National Academies of Sciences, Engineering, and Medicine, "Global change research needs and opportunities for 2022-2031," National Academies Press, Washington, DC, USA, 2021. [Online]. Available: https://www.nap.edu/catalog/26055/global-change-research-needs-and-opportunities-for-2022-2031
2. National Academies of Sciences, Engineering, and Medicine, "Pathways to discovery in astronomy and astrophysics for the 2020s," National Academies Press, Washington, DC, USA, 2021. [Online]. Available: https://www.nap.edu/catalog/26141/pathways-to-discovery-in-astronomy-and-astrophysics-for-the-2020s
3. "Transforming science through cyberinfrastructure: NSF's blueprint for a national cyberinfrastructure ecosystem for science and engineering in the 21st century," National Science Foundation, Alexandria, VA, USA, 2019. [Online]. Available: https://www.nsf.gov/cise/oac/vision/blueprint-2019/

4. "Pioneering the future advanced computing ecosystem: A strategic plan," National Science and Technology Council, Washington, DC, USA, 2020. [Online]. Available: https://www.nitrd.gov/pubs/Future-Advanced-Computing-Ecosystem-Strategic-Plan-Nov-2020.pdf
5. A. Blatecky *et al.*, "The missing millions: Democratizing computation and data to bridge digital divides and increase access to science for underrepresented communities," National Science Foundation, Alexandria, VA, USA, 2021. [Online]. Available: https://www.rti.org/publication/missing-millions/fulltext.pdf
6. "Transforming science through cyberinfrastructure: Coordination services: NSF's blueprint for national cyberinfrastructure coordination services for accelerating science and engineering in the 21st century," National Science Foundation, Alexandria, VA, USA, 2019. [Online]. Available: https://www.nsf.gov/cise/oac/vision/blueprint-2019/nsf-aci-blueprint-services.pdf
7. "NSF ACCESS awardees will advance innovations in cyberinfrastructure accessibility, user support and integration services," National Science Foundation, Alexandria, VA, USA, 2022. [Online]. Available: https://nsf.gov/news/special_reports/announcements/042222-access.jsp
8. "Dear colleague letter: Pilot for the allocation of high-throughput computing resources (HTC)," National Science Foundation, Alexandria, VA, USA, 2022. [Online]. Available: https://www.nsf.gov/pubs/2022/nsf22051/nsf22051.jsp
9. "Transforming science through cyberinfrastructure: NSF's blueprint for cyberinfrastructure learning and workforce development," National Science Foundation, Alexandria, VA, USA, 2021. [Online]. Available: https://www.nsf.gov/cise/oac/vision/blueprint-2019/CI-LWD.pdf
10. "Transforming science through cyberinfrastructure: NSF's blueprint for a national cyberinfrastructure ecosystem for science and engineering in the 21st century," National Science Foundation, Alexandria, VA, USA, 2021. [Online]. Available: https://www.nsf.gov/cise/oac/vision/blueprint-2019/CI-LWD.pdf
11. "Dear colleague letter: Nurturing diverse, skilled, capable, and productive communities of cyberinfrastructure professionals (CIP)," National Science Foundation, Alexandria, VA, USA, 2022. [Online]. Available: https://beta.nsf.gov/funding/opportunities/nurturing-diverse-skilled-capable-and-productive-communities
12. "Training-based workforce development for advanced cyberinfrastructure (CyberTraining)," National Science Foundation, Alexandria, VA, USA, 2022. [Online]. Available: https://beta.nsf.gov/funding/opportunities/training-based-workforce-development-advanced-cyberinfrastructure
13. "CI CoE: Demo pilot: Advancing research computing and data: strategic tools, practices, and professional development," National Science Foundation, Alexandria, VA, USA, 2021. [Online]. Available: https://www.nsf.gov/awardsearch/showAward?AWD_ID=2100003
14. "RCD nexus," Campus Research Computing Consortium, 2021. [Online]. Available: https://carcc.org/rcd-nexus/
15. "Minority Serving - Cyberinfrastructure Consortium." https://www.ms-cc.org
16. "The National Artificial Intelligence Research Resource Task Force," National Artificial Intelligence Initiative Office, 2020. [Online]. Available: https://www.ai.gov/nairrtf/

MANISH PARASHAR is the office director of the Office of Advanced Cyberinfrastructure, National Science Foundation, Alexandria, Virginia 22314, USA. Contact him at mparasha@nsf.gov.

EDITOR: Phil Laplante, The Pennsylvania State University, plaplante@psu.edu

DEPARTMENT: SOFTWARE ENGINEERING

This article originally appeared in
Computer
vol. 56, no. 3, 2023

The Byzantine Empire and Its Generals: An Ancient Empire Back to Life in Computer Security

Pedro Reviriego, *Universidad Politécnica de Madrid*
Elena Merino-Gómez, *Universidad de Valladolid*
Fabrizio Lombardi, *Northeastern University*

Forty years after its initial publication, we revisit the seed contribution of Byzantine fault tolerance, focusing on its application for the security of systems implemented in software. We describe new environments in which it is being used.

Security and dependability are key requirements for most of today's computing systems, and their importance is poised to grow as we increasingly rely on their pervasive use in almost every aspect of our lives. At the same time, the complexity of computing systems unabatedly continues to grow with many different organizations providing interdependent components that, in turn, coordinate to implement services. In this scenario, making sure that such a system will work reliably when some of the components or nodes fail or are compromised by an attacker becomes critical. For example, several attacks, like Spectre or Meltdown, have been investigated; they exploit the advanced mechanisms of modern processors to extract information. Similarly, in the recent SolarWinds attack, a software tool was compromised, and then automatic updates were exploited to disseminate the infected version. Failures can also disrupt the operation of computing systems in many complex ways. For example, radiation-induced soft errors can flip any bit stored in a memory or register, leading to silent data corruption that can manifest in erratic system behavior.

From a design perspective, in many cases, the same mechanisms can be used to mitigate both attacks and failures. In fact, many of the models commonly used for secure and dependable system design cover both scenarios. This is the case with the Byzantine Generals problem formulated more than 40 years ago; this has led to the concept of Byzantine fault tolerance (BFT), which has found widespread adoption in many technical domains.[1]

In this article, we revisit BFT four decades after its introduction, focusing on software implementations and briefly discussing how it is now being used in new systems, domains, and applications. We also look back to the Byzantine empire to understand how it survived for one millennium and how its history relates to the Byzantine generals problem. This discussion links computing with history and shows that the choice made by the authors for the generals is, in an unintended way, backed by facts.

An analogy with a group of generals who have to act consistently in taking the decision to attack or retreat has been used in Lamport et al.[1] to provide a model for secure and dependable system design. The generals can be loyal or traitors, and there can also be communication failures or restrictions among generals. This models a computing system in which some nodes may have been compromised and in which failures could also either disable nodes or prevent them from communicating.

As has been the case with other famous problems in computing, the analogy can facilitate the understanding of the problem and the algorithms used to solve it. Indeed, formulating the problem with an

Digital Object Identifier 10.1109/MC.2023.3235095
Date of current version: 8 March 2023

 Published by the IEEE Computer Society

appealing analogy was one of the objectives of the authors of the article [see http://lamport.azurewebsites.net/pubs/pubs.html#byz (46. The Byzantine Generals Problem)]. Apparently, they chose the generals to be Byzantine to avoid offending any nationality, and thus, the model became the Byzantine Generals problem.[1] From then on, systems and algorithms that can solve this problem and work consistently in that scenario are known as Byzantine fault tolerant (BFT). Hence, BFT has become a key concept in dependable and secure system design.

After presenting the initial model, different scenarios, for example by considering that messages exchanged by the generals can be forged by traitors or, conversely, that they are signed and thus cannot be forged, are analyzed in Lamport et al.[1] This has led to a fundamental result; for the group of loyal generals to act consistently, there can be at most m traitors in a group of $3m + 1$ generals when messages can be manipulated by traitors. For example, when there is a single traitor, there have to be at least three loyal generals for them to act consistently. This illustrates the high cost of building systems that can tolerate failures or attacks; not only are $3m + 1$ generals needed, but they must also exchange a sequence of messages recursively to reach a consensus on the action to take.

By presenting the problem in a general manner, considering different scenarios with signed or oral messages and with failures or restrictions in the communications among the generals, the article instantiated a framework for the analysis and design of fault-tolerant systems that has been used in a myriad of applications and designs. Initially, the concept was used for safety-critical applications such as space systems; avionics; military equipment; or industrial and nuclear control systems. However, its adoption has extended to almost every domain in computing. For example, BFT is a key element in many blockchain-based systems, and in particular for cryptocurrencies, to ensure that a group of completely

FIGURE 1. The Ethereum logo.

independent nodes can maintain a consistent state. This has motivated a large body of research throughout the years in this area to ensure that consensus can be achieved reliably in systems that involve large numbers of nodes and transactions. These efforts have led to the development of new consensus algorithms, such as, for example, proof of work and proof of stake.[2]

The game with the nationality of the generals seems to continue. For example, three of the main forks in Ethereum (see Figure 1) are named *Byzantium*, *Constantinople*, and *Istanbul*, which does not seem to be a coincidence, and it is likely a play with words and a tribute to the Byzantine generals problem.

Before discussing other areas in which BFT is currently being used, let us go back in time and look at the empire that gives the name to the problem and concept. For more than one millennium, the Byzantine empire was able to survive despite having to face powerful enemies from the East and West. Therefore, in a way, the empire can itself be seen as a complex resilient system from the outset. Most of its rulers had a solid military background; lineage was not a sufficient condition—sometimes not even necessary. For emperors and coemperors to be considered worthy to wear the imperial purple, they had to have demonstrated their ability as generals, which partially explains the strategic strength the empire enjoyed for centuries. Right from the start, the binomial formed by

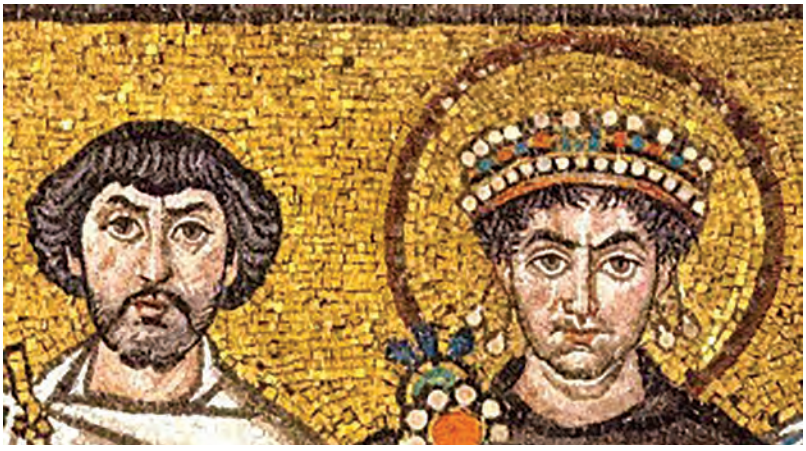


FIGURE 2. A portrait possibly of General Belisarius and Emperor Justinian (from a mosaic in San Vitale, Ravenna.)

FIGURE 3. The iron chain prevents the fleet of Thomas the Slav from entering the Golden Horn. (Source: Biblioteca Nacional de España.)

Justinian and his general Belisarius (Figure 2) paved the way that would forever associate military power with imperial power, often in a single person, a circumstance that would be essential in defensive strategies.

In addition to the skill of many of its rulers, defensive resources, such as the chain of the Golden Horn (Figure 3) or the one known as *Greek fire* (Figure 4), were fundamental for the Byzantine resistance to attacks throughout its history. It is significant that even in the case of war tools from the past, they continue to pose enigmas in the present. The chemical composition of Greek fire, a kind of liquid fire capable of spreading over water and easily reaching enemy ships, has not yet been formulated. The precise mechanism that allowed the closing of the Golden Horn by means of a heavy chain that was pulled up to the surface is even today the subject of speculation. In addition to the scientific challenge of unraveling how these tools worked, their efficient performance can be inspiring for engineers who are currently developing cyberdefense systems.

The adoption of the term *Byzantine* to describe the problem of the Byzantine Generals is a cautious decision that contrasts with the apparently inappropriate use to denominate the Eastern Roman Empire. Leslie Lamport assigned the Byzantine nationality so as not to offend any reader in the certainty that an extinct empire was a safe bet. However, the name "Byzantine" turned out to be one of the most controversial of the historical empires. Curiously, the Byzantines themselves would surely have felt bothered with the name attributed to them to avoid calling them "Romans," as they considered themselves. The name "Byzantine" to name the Eastern Roman Empire is subsequent to the disappearance of the empire itself. The origin of the name and its connotations are still part of a scientific discussion today.[3]

Lamport also confesses that he took the idea of the generals from the problem in distributed computing that is sometimes called the *Chinese Generals problem*, "in which two generals have to come to a common agreement on whether to attack or retreat, but can communicate only by sending messengers who might never arrive."[1] The idea of going back to the past arises intuitively at the moment in which the figure of the messenger appears. The times when

messages could be transmitted only through intermediaries seem to contrast with the current situation in which the immediacy of sending and receiving can give the impression that the endpoints of the communication are enough. However, it is obvious, although imperceptible, that the messenger, which connects the sender and receiver, continues to be present, merged, or frequently confused with the channel.

The immediacy of the transmission of information, so common nowadays, can lead to a feeling of false security in the minds of communicators. It is possible to mistakenly perceive that the instantaneity and the apparent absence of intermediaries guarantee the veracity of the message. The speed of communication does not seem to offer enough time to intentionally alter the message. In a similar way, immediacy seems to establish a direct thread with the addressee, without intermediaries. However, messages and the channels through which they travel can be as insecure today as they were 500 years ago. The messages today must go through a myriad of hardware and software components and systems before reaching their destination. Not only can the senders or recipients of the message be malicious, but all those complex elements can also be manipulated to interfere with and disrupt communication, opening a vast attack surface. In fact, the security of communications has been a critical issue since the beginning of its existence and is one of the oldest problems in the history of communication.

FIGURE 4. An illustration of Greek fire, from J. Scylitzes (flourished), *History of Byzantium*. (Source: Biblioteca Nacional de España.)

FIGURE 5. A large miniature depicting a view of besieged Constantinople from Jacques Tedaldi, *Recueil de textes historiques Récit de la prise de Constantinople* (1453). (Source: Bibliothèque nationale de France.)

The Byzantine Empire offers heroic examples of safe message delivery. One of the most thrilling episodes in the transmission of a message occurred only a few days before the Fall of Constantinople (see Figure 5). Constantine XI Palaiologos, the last Byzantine emperor, urgently needed to know if more reinforcements from Venice would arrive in Constantinople. Without them, the city was doomed. Twelve men trusted by Constantine embarked in a small brigantine, with a false flag, disguised as Turks, toward the Aegean Sea, to see if the necessary help

was approaching. The 12 messengers found that there was no help on the horizon. They were to return to Constantinople to deliver their hopeless message. One of the messengers proposed to the others to continue toward Christian lands to save their own lives. Returning to Constantinople to deliver the message to the emperor meant certain death. However, loyalty prevailed, and they returned to the city to transfer the information even at the cost of their lives.[4] Although the precious message arrived uncorrupted on 23 May 1453, it was already too late for everything, and just six days later, the city would fall, and with it, the last stronghold of the empire.

Although the error-free transmission of this last message could do nothing to prevent the final Fall of Constantinople, throughout the history of the Byzantine Empire, there are other kinds of episodes that exemplify how the system was able to survive despite the spread of erroneous messages. One of the versions of the events that occurred in the famous Battle of Manzikert in 1071, between Byzantines and Seljuk Turks, illustrates how false messages about the defeat of Emperor Romanus IV Diogenes were spread. It is possible that the jealous Byzantine general Andronikos Doukas, belonging to the family that had ruled Byzantium in the previous generation, took advantage of the confusion in the transmission of a message to abandon the emperor in battle.

AFTER 40 YEARS, SECURITY AND DEPENDABILITY HAVE BECOME CRITICAL DESIGN REQUIREMENTS, AND BFT HAS BEEN USED IN A MYRIAD OF SYSTEMS, DOMAINS, AND APPLICATIONS.

Romanus IV Diogenes gave the signal to bring the pursuit against the Turks to a halt,[5] fearing an ambush. The message was misinterpreted in the rear guard. The order to return to the camp was interpreted as a withdrawal, and it was deduced that the emperor had fallen in his advance against the enemies. Some argue that the rumor was actually started by Andronikos Doukas, who did not forgive Romanus IV for having interrupted the succession of the house of Doukas to the throne of Byzantium. The contaminated message of the fall of Emperor Romanus IV Diogenes caused the abandonment of his people and determined that, indeed, he was finally captured by the Seljuk Sultan Alp Arslan. Although the defeat of Manzikert, due in part to the spread of an adulterated message, is considered as one of the greatest disasters in the history of the empire, the Byzantines continued to persist.

Seven decades later, the transmission of a corrupted message once again put another emperor in trouble. During the maneuvers to recapture Antioch for the Byzantine Empire, Emperor John II Komnenos was betrayed by two apparent allies: Raymond of Antioch and Joscelin II, Count of Edessa. The latter sent secret messengers to spread the false message to the citizens of Antioch that Emperor John II Komnenos wanted to harm them. The rumor that Antioch had been sold to the Byzantine Greeks and that the citizens should leave their homes forced one of the greatest emperors in the history of Byzantium to leave Antioch in 1142.[6] However, the empire still had more than three centuries to live.

The problem of the Byzantine Generals is still valid 40 years later, and its name, even if it was adopted to avoid potentially more controversial terms, demonstrates its relevance in the face of the many examples that can be extracted from the long history of Byzantium. It would also be interesting to consider some of the weak points of Byzantine history to name possible security flaws. For example, according to Doukas, a contemporary historian of the Fall of Constantinople, the door next to the circus, known as the *kerkoporta*, was left ajar, and 50 Janissaries slipped through the unattended door.[7] The chapter of the forgotten door could have been decisive for the final blow to the city on 29 May 1453. This is very similar to backdoors that are created to gain unauthorized access to computing systems today.

However, different from the Byzantine Generals problem, the link with the past was not made, thus losing the opportunity to use the term "kerkoporta" to denote security backdoors, and thus, keeping the term for the collective memory of humanity. However, there may be some hope as recently "kerkoporta" has been used to name a ransomware, so in the long run, the term may be adopted, increasing the links between the old Byzantine empire and computer security.

After 40 years, security and dependability have become critical design requirements, and BFT has been used in a myriad of systems, domains, and applications. In fact, in recent years, new scenarios for BFT have emerged. For example, due to storage and processing limitations or privacy concerns, machine learning is increasingly being implemented in multiple nodes. Typically, each node stores or generates a part of the dataset, and all nodes cooperate to implement training or inference. For example, distributing the dataset among several computing nodes in a data center can provide large speedups for training, while in Internet of Things applications, the nodes commonly operate in a decentralized manner with limited capability to exchange data.[8]

The use of several nodes creates the need for the system to operate reliably when some of the nodes fail or are compromised. A good example is federated learning, which is emerging as a technology that can enable learning from many users or sensors while preserving privacy. Basically, training is done locally without sharing the data, and the results from many devices are aggregated to obtain a model based on data from all of them. The implementation of federated learning poses challenges to developing efficient algorithms to coordinate training but also to ensure that it is robust when some of the devices fail or act maliciously. Therefore, there is a strong need to implement BFT at scale in federated learning. Different schemes have been proposed; they try, for example, to detect the updates from malicious nodes by comparing them with those of the rest of the nodes or to reduce their impact on the aggregated result.

Distributed nodes or sensors are used not only for training but also for inference, and then again, there is a need to make sure that the system can withstand the failure or misbehavior of some of them. This can be achieved by carefully analyzing the information coming from each sensor to estimate their reliability and use them accordingly for the inference process.[9] Therefore, the trend to use distributed systems to implement both training and inference makes BFT a key element for future machine learning systems.

Networking is another area in which BFT is becoming increasingly important. For example, in software-defined networks, controllers are critical, and thus, they are typically replicated to tolerate failures. As security is also a major issue in networks, providing BFT for the control plane when some of the controllers may have been compromised is also desirable in all cases and needed for networks used for critical applications. Several schemes have been proposed to reduce the overhead of implementing BFT by first identifying the disagreement among a subset of controllers and only then activating all the controllers needed to implement BFT or to detect malicious controllers.[10]

Similarly, BFT is also fundamental in wireless sensor networks that are by nature decentralized systems and for which attackers can use sophisticated mechanisms or direct jamming to disrupt communications. The ability to send broadcast messages in real time is also critical in some systems, and thus, BFT has to be implemented.[11] In summary, networks are by nature distributed, and thus, they can suffer failures and compromised nodes, thus making BFT imperative when reliable operation is needed.

Distributed optimization, similarly to distributed machine learning, relies on different nodes to optimize a function; this can be done locally and independently at each node or in a coordinated way.[12] In all cases, there can again be faulty or malicious nodes, and thus, there is a need to implement BFT. New mechanisms to support the coordination of multiple agents to perform a given computation with BFT are being proposed[13] by using replication; such a scheme can be used as a general solution when the cost introduced by replication is acceptable.

The use of quantum technologies has also been proposed to reach an agreement between generals.[14] Soon, with computing systems moving toward more complex, distributed, and in many cases, decentralized systems, the importance of BFT is poised to keep growing. Therefore, these first 40 years seem to be only the beginning of a new Byzantine era, but this time in computer science rather than as an empire.

Looking forward, we think that using analogies when presenting new algorithms and ideas can be a powerful tool to catch the attention of the readers, enable a formulation of problems and solutions that is more general, and link computer science with other fields like history. The Byzantine Generals problem is an excellent example of how those benefits can be

achieved. However, that has not always been the case. In fact, Leslie Lamport used another analogy to describe a consistency algorithm, relating it to an ancient parliament formed by part-time legislators in the Greek island of Paxos, but in this case, it seems that, at least initially, the analogy was not well received [see http://lamport.azurewebsites.net/pubs/pubs.html#lamport-paxos (123. The Part-Time Parliament)]. Therefore, as with any powerful tool, analogies should be used with caution.

REFERENCES

1. L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, Jul. 1982, doi: 10.1145/357172.357176.
2. A. Bessani, E. Alchieri, J. Sousa, A. Oliveira, and F. Pedone, "From Byzantine replication to blockchain: Consensus is only the beginning," in *Proc. 50th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, 2020, pp. 424–436, doi: 10.1109/DSN48063.2020.00057.
3. A. Kaldellis, "From Rome to New Rome, from Empire to Nation-State," in *Two Romes: Rome and Constantinople in Late Antiquity*, L. Grig and G. Kelly, Eds. London, U.K.: Oxford Univ. Press, 2012, pp. 387–404.
4. N. Barbaro, *Giornale Dell'assedio di Constantinopoli*. Vienne, France: Libreria Tendler, 1856, p. 35.
5. M. Attaleiates, *The History*. (Transl.: A. Kaldellis and D. Krallis, Dumbarton Oaks Medieval Library 16). Cambridge, MA, USA: Harvard Univ. Press, 2012, p. 293.
6. W. of Tyre, *A History of the Deeds Done Beyond the Sea*, vol. 2. New York, NY, USA: Columbia Univ. Press, 1943, p. 97.
7. M. Philippides and W. K. Hanak, *The Siege and the Fall of Constantinople in 1453: Historiography, Topography, and Military Studies*. Burlington, VT, USA: Ashgate Publishing, 2011, p. 622.
8. Z. Yang, A. Gang, and W. U. Bajwa, "Adversary-resilient distributed and decentralized statistical inference and machine learning: An overview of recent advances under the byzantine threat model," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 146–159, May 2020, doi: 10.1109/MSP.2020.2973345.
9. J. Choi, Z. Hakimi, J. Sampson, and V. Narayanan, "Byzantine-tolerant inference in distributed deep intelligent system: Challenges and opportunities," *IEEE Trans. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 3, pp. 509–519, Sep. 2019, doi: 10.1109/JETCAS.2019.2933807.
10. E. Sakic, N. Đerić, and, and W. Kellerer, "MORPH: An adaptive framework for efficient and Byzantine fault-tolerant SDN control plane," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2158–2174, Oct. 2018, doi: 10.1109/JSAC.2018.2869938.
11. D. Kozhaya, J. Decouchant, and P. Esteves-Verissimo, "RT-ByzCast: Byzantine-resilient real-time reliable broadcast," *IEEE Trans. Comput.*, vol. 68, no. 3, pp. 440–454, Mar. 1, 2019, doi: 10.1109/TC.2018.2871443.
12. L. Su and N. H. Vaidya, "Byzantine-Resilient multiagent optimization," *IEEE Trans. Autom. Control*, vol. 66, no. 5, pp. 2227–2233, May 2021, doi: 10.1109/TAC.2020.3008139.
13. R. Guerraoui and A. Maurer, "Byzantine-resilient multi-agent system," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 6, pp. 4032–4038, Nov./Dec. 1, 2022, doi: 10.1109/TDSC.2021.3116488.
14. M. Fitzi, N. Gisin, and U. Maurer, "Quantum solution to the Byzantine agreement problem," *Phys. Rev. Lett.*, vol. 87, no. 21, Nov. 2001, Art. no. 217901, doi: 10.1103/PhysRevLett.87.217901.

PEDRO REVIRIEGO is an associate professor at the Universidad Politécnica de Madrid, 28040 Madrid, Spain. Contact him at pedro.reviriego@upm.es.

ELENA MERINO-GÓMEZ is an assistant professor at the Universidad de Valladolid, 47011 Valladolid, Spain. Contact her at elena.merino.gomez@uva.es.

FABRIZIO LOMBARDI is the International Test Conference Endowed Chair Professor at Northeastern University, Boston, MA 02215 USA. Contact him at lombardi@coe.neu.edu.

Get Published in the New *IEEE Open Journal of the Computer Society*

Submit a paper to the new IEEE Open Journal of the Computer Society covering computing and informational technology.

Your research will benefit from the IEEE marketing launch and 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.

IEEE COMPUTER SOCIETY | IEEE

EDITOR: Colette Perold, colette.perold@colorado.edu

DEPARTMENT: THINK PIECE

This article originally appeared in
IEEE Annals of the History of Computing
vol. 43, no. 4, 2021

Mathematics, Logic, and Engineering in Computing

Peter J. Denning, *Naval Postgraduate School, Monterey, CA, USA*
Matti Tedre, *University of Eastern Finland, Joensuu, Finland*

From its beginning in the 1950s, noncomputing academics were skeptical about computer science because it seemed strong on technology and weak on theory. To answer the critics and shore up their case, computer scientists turned to a rich trove of computational methods from logic and mathematics. Computing from ancient times focused on methods of manipulating symbols that could be performed by people untrained in mathematics. Examples include ancient Babylonian algorithm-like step-by-step rules, Greek mathematical procedures like the Euclidean algorithm or the sieve of Eratosthenes, and al-Khwarizmi's algorithmic techniques. In the twentieth century, the mathematical logicians Turing, Gödel, Church, Kleene, and Post provided a solid foundational theory for the new field of computer science, showing what can and cannot be computed.

Much human computation is based on procedures of many steps, many of which depend on logic: decomposing a large task into a series of smaller ones, choosing between alternative tasks based on a condition, and repeating tasks until some condition was achieved. Popular history texts in computing cite the development of logic as a means for ultimately automating these choices. Boole's algebra for logic formulas (1854) gave a notation for conditions used in making the choices, which could be composed from simple true–false elements connected by AND, OR, and NOT. Shannon's insight (1938) made Boole's algebra the basis for describing electronic computer circuits. Frege's axiomatic predicate logic (1879) presented formal rules of inference and syntax, which, in the 1950s, came to be seen as the logical basis for programming languages. Like many others, Boole and Frege believed that logic was the foundation for rational human thought.

The notion that logic enabled computation opened the reverse possibility that computation could automate the logic of mathematical proofs. Beginning around 1900, prominent mathematicians and logicians sought to characterize the process of proof so precisely that an automatic procedure could decide whether any given proposition is provable in first-order logic. In 1928, the famous mathematician David Hilbert posed this Decision Problem as one of the fundamental challenges in mathematics [5].

To many, the Decision Problem looked eminently doable: a proof system consisted of given axioms and rules of inference, and a proof is a well-structured sequence of statements in which each statement is either an axiom or is constructed from previous statements by a rule. Hilbert and many others had believed for years that an algorithm for the Decision Problem existed, although they never could find it. Their hopes were permanently dashed when, in the 1930s, Gödel, Post, Turing, and Church proved that this was impossible. Their different systems of mechanization were all shown to be equivalent—any computation in one could be simulated in all the others. The famous Church–Turing thesis stated that all effective computations could be formalized as Turing machines. What an irony, that logic-inspired computation was incapable of answering the Decision Problem's question of whether logical proofs could be automated.

This irony supported the academic case for computer science. Not only did the work of Turing and the others provide a basic theory of computation, it led to the surprising conclusion that many important questions cannot be answered by computational algorithms. It made the case that a small set of logic principles governed the thought processes of designing algorithms.

For these reasons, when computing became an academic discipline in the 1950s, the popular disciplinary narratives of computing prominently featured mathematics and logic.

COMPUTING AS A BRANCH OF LOGIC

Many pioneers of the nascent computing field in the 1950s came from mathematics. They took it as a given

1058-6180 © 2022 IEEE
Digital Object Identifier 10.1109/MAHC.2021.3132110
Date of current version 7 February 2022.

 Published by the IEEE Computer Society

that electronic automatic computers are governed by logic systems. Engineers agreed: logic was baked into computer architecture from the beginning. In his 1938 M.Sc. thesis, Claude Shannon showed that the functions of switching circuits, such as those found in telephone exchanges and motor control equipment, could be described by Boolean algebra. Through a series of developments into practical applications, Shannon's insight permeated the engineering world [1] and was eventually adopted for computer circuits. Electronic computer circuits came to be called logic circuits, and Boole's algebra of logic became the standard basis for computing. Logic's influence on circuit design continued in the 1950s and 1960s. For instance, an early design technique using Karnaugh maps (1953) enabled logic circuit designers to minimize the number of logic gates needed and avoid race hazards where a change of state could cause an output to flicker. Logic also influenced programming practice. The 1966 Böhm–Jacopini theorem restated the logic basis of programming languages: every computation could be constructed from simpler computations by joining them in sequences, if-then clauses, or iteration clauses. The theorem was taken as a support for "structured programming"—a programming practice that advocated a limited set of constructs to build programs.

Similarly, early research programs in artificial intelligence were based on an idea that human intelligence (at least the rational part) is based on logic. This idea was celebrated in the monumental intellectual achievements of the early 1900s, notably Russell and Whitehead's *Principia Mathematica* and Wittgenstein's *Tractatus Logico-Philosophicus*. Logic was revered as a pinnacle of human intelligence. Not surprisingly, early AI focused on getting logic programs to perform intelligent actions. The logic theory of intelligence received a big boost in 1956, when the *Logic Theory Machine* of Newell, Simon, and Shaw proved 38 of the first 52 theorems of the *Principia*. Some saw that machine as an improvement to human intelligence—it produced in a few minutes proofs of several theorems that brilliant thinkers took years to prove. Logic came to be seen as a litmus test for machine intelligence.

This idea spawned a branch of AI devoted to logic programming, from which expert systems emerged in the early 1980s. The prototypes used new logic languages LISP and PROLOG. Engineers built special-purpose machines to run programs in these languages very efficiently. The Japanese Fifth Generation Project (1980s) was aimed at turbocharging expert systems by building supercomputers for massive logic operations, just as a numerical supercomputer could do with massive arithmetic operations. The United States responded with its Strategic Computing Initiative, more generally focused on supercomputers capable of solving "grand challenge problem" in science.

Logic pervaded other parts of computing as well. In 1970, Codd introduced the logic of relational databases, which became a major IBM project later in that decade and spawned a host of database companies. These systems are often queried and managed with the language SQL, which consists of logic expressions to select, join, and project records. Logic made the data management systems common in business simpler and more effective.

In the same time, there was an explosion of insights into the complexity of computations. It was well known that some problems are harder to solve than others—their algorithms take more time and memory. The NP-completeness theorems of Cook (1971) and Levin (1973) to characterize these problems are deeply rooted in logic. Many hard decision problems could be simulated with gigantic logic circuits; finding the answer amounted to finding an input to the logic network that produces the desired yes–no output. This is known in logic as the satisfiability (SAT) problem. Any fast algorithm for solving the SAT problem would be convertible to a fast algorithm for any of the numerous hard problems that could be simulated as a SAT problem. Our theory of complexity rests on logic.

CRACKS APPEAR

Yet, logic's hegemony had already started to show signs of cracking in the late 1960s. When software engineering was being born to address the software crisis, several pioneers from the logic-oriented community proposed that much of the unreliability of software would be eliminated if the software could be formally proved to meet its specifications, for then there would be no doubt that the software was error free once it was compiled. Formal verification, however, turned out to be a formidable challenge. It sparked heated debates that forever shaped the computing field.

It turned out that formal logic proofs could be carried out only for relatively small programs, but large systems were beyond their reach. Even if the program source code could be proved correct, there was the additional difficulty of proving that the compiled machine code as well as the hardware platform also met their specifications. In his 1983 Turing Award lecture, Ken Thompson reminded us that bugs were not a feature of program code alone but of the total software-hardware–human system. Logicians and engineers argued endlessly about the practicality of formal proof. Many engineers were concerned that recovery from defects and deterioration of hardware—such as a transistor failure or arrival of a signal corrupted by noise—could not be supported in the

logic formalisms. The prospects for full and complete verification started to look very gloomy when many agreed that developing complete specifications representing the true intentions of the human stakeholders and users of systems could not be formalized—and therefore not amenable to the tools of standard logic.

Other anomalies about the completeness of logic as the basis of computing appeared. In the mid-1960s, Lotfi Zadeh argued that engineers are frequently faced with ambiguous situations where a condition can be partially true and partially false at the same time. He proposed "fuzzy logic" as an extension of Boolean logic that allows truth values to be represented by numbers between 0 and 1. Fuzzy logic proved useful in many physical devices, but did not gain much foothold in the AI and logic community. Still another anomaly in the field of AI was found with expert systems, which were expected to perform as a human expert by acquiring enough deductive rules and facts. Dreyfus (1972) challenged this idea on the grounds that much expert behavior does not follow known rules; expert systems might become competent, he argued, but not expert [4]. Many other anomalies between the expectations of what AI could achieve and what AI actually achieved arose from the presumption that intelligence is founded in formal logic.

In 2013, Moshe Vardi lamented about the accumulation of gloomy conclusions about logic two decades before. He recalled how he and his colleagues experienced a feeling that large-scale program verification may indeed be hopeless: "First-order logic is undecidable, the decidable fragments are either too weak or too intractable, even Boolean logic is intractable [6]." Moreover, as computers invaded many new areas such as entertainment, cyber-physical control systems, office tools, art, transportation, medicine, and more, skillful development relied progressively less on logic and more on design acumen, human communication, aesthetics, social savvy, and other nonformal skills. Continuing progress in important technologies such relational databases, Boolean reasoning, and model checking did not stave off the growing feeling that computing could not be reducible to logic.

NO COMPUTING WITHOUT ENGINEERING

Over the 1980s, there was a growing consensus that the logic view does not cover many engineering, science, and technology aspects of computing. We were being pulled back to the historic notion that the roots of computer science are a complex mixture combining mathematics, science, and engineering. Logic did not cover everything in computing. The 1989 *Computing as a Discipline* report crystallized this growing feeling among people in the computing field [3]. Herewith a few examples.

Start with logic circuits. The logic formulas describing circuits assume that the signals are 0 and 1. But these are abstractions. The 0 and 1 represent states of the circuits, such as voltage low or high. Because a physical circuit can be in transient states that are neither 0 nor 1, the logic of the abstraction is unable to deal with some physical behaviors. The "half signal problem" asks what happens when one part of a circuit tries to read another part that has not settled into a definite 0 or 1 state. The "arbitration problem" asks what happens if a logic signal and clock signal arrive at the input of a flip-flop circuit at the same time. This condition can trigger the flip-flop into a metastable state that is neither 0 nor 1 and can crash the CPU when it persists for many clock ticks. These problems have physical solutions that cannot be derived in logic. Physical circuits display important stochastic behaviors that cannot be addressed by logic alone.

Next, consider the architecture of computers. In 1945, John von Neumann published the ideas of a team of pioneers from the early computing projects who defined a better architecture that would be more reliable and faster than their previous machines. What emerged is now known as the von Neumann architecture. It separated the computer into CPU, memory, and input–output, and defined the CPU cycle that fetches and executes programs stored as instructions in the main memory. While this architecture has often been held as an example of logical and analytic thinking in computing, it was actually the product of engineering improvements for efficiency and reliability [2].

One of the innovations of that architecture was to fetch instructions from main memory rather than paper tapes or cards. This engineering innovation greatly speeded up program execution. However, folklore developed that the stored program idea was the implementation of Turing's universal machine. This is not so. Historians find evidence to the contrary that the architecture was not influenced by Turing's model, nor did Turing have in mind an architecture of the same type. Other folklore held that the new architecture would be easier to build than its predecessors. In some ways, it surely was, but at the same time it also created new challenges. For instance, Maurice Wilkes, who led the EDSAC project at the University of Cambridge, said that one of the many engineering challenges was finding a technology that could support a main memory large enough to hold all the instructions of the program. Wilkes found that a mercury delay line did the job better than other available

technologies. With this and other engineering innovations, he got the EDSAC machine working a year earlier than its U.S. version (EDVAC). Wilkes steadfastly maintained that there are important aspects of computing that logic cannot address.

Next, consider large multiuser networked systems. The Multics project at MIT (1965) dreamt of a "computer utility" that would dispense cheap computing power over a network to the masses. No one knew how to organize large operating systems that would coordinate hundreds of users. The logic-inspired view at the time was to carefully construct the operating system as a set of modules that interacted by well-defined interfaces. But these systems had great difficulties with coordination and were prone to many errors and crashes. Operating systems designers at MIT, IBM, and elsewhere invented a new idea, the process, as the basic entity demanding service from the system, and they organized the system as a "society of cooperating processes." This led rapidly to successful operating systems and a new theory of concurrent process coordination. Although logic helped to make coordination theories more precise, neither the gestation of the process idea nor its development was in logic—the process arose in the pragmatics of coordinating activities in an operating system. Over the years, the engineering understanding of these systems led to very small operating system kernels that could be formally verified by the methods pioneered in the 1970s. Today the sel4 secure operating system kernel illustrates how an engineered system can progress to the point of being a commercially viable, fully verified kernel.

Computational science, which grew up in the 1980s, is based on the idea that many physical processes can be viewed as information processes that can be simulated on a computer. Formal logic does not capture the simulation and modeling prevalent in computational science.

Finally, consider the performance of computer systems. As users, we want computers to get our jobs done as fast as possible within the constraints of processors and memory. Complexity theory gave order-of-magnitude estimates of running times of algorithms on a single CPU. But it was not able to predict the response time when multiple jobs were competing for the CPU. The solution to this was again found by engineers who recognized that queueing theory could answer the question. Computer scientists plunged into the performance analysis and prediction problem and discovered very fast algorithms to compute throughput and response times for operating systems (and the Internet) built as networks of servers. This led eventually to a thriving performance-evaluation industry. But queueing theory is not a product of logic, and performance evaluation is an empirical, not logical, matter.

In all these systems, engineering was oriented toward finding what works and what does not work. This is often accomplished with lots of trial and error, tinkering, and experimenting. There is often no theory or science available to understand what is going on; understanding is developed by trying things out. For instance, designers of early time-sharing systems found no theory that could predict their response time. Once time-sharing systems started to show promise, a rich body of theory emerged to accurately predict response time, guide the design of systems, and evaluate their performance.

The modern computational thinking movement for K-12 education has embraced the idea that computational thinking is founded on logical thinking. The movement has defined curricula that teach computing principles using generic logic puzzles and games. This has been controversial because generic logic does not demonstrate the unique aspects of computing and because it omits study of engineering and design in computing systems.

In truth, computer science is built on a complex framework of understandings from logic, mathematics, science, and engineering. Combined together, these different modes of thinking produced the amazing progress we have seen in computing. Computing is bigger than logic, and logic is less foundational than many people believe: designing, building, experimenting, and are at least equally foundational for the field.

REFERENCES/ENDNOTES

[1] L. De Mol, M. Bullynck, and E. G. Daylight, "Less is more in the fifties: Encounters between logical minimalism and computer design during the 1950s," *IEEE Ann. Hist. Comput.*, vol. 40, no. 1, pp. 19–45, Jan.–Mar. 2018.

[2] L. De Mol and M. Bullynck, "Roots of 'Program' revisited," *Commun. ACM*, vol. 64, no. 4, pp. 35–37, 2021.

[3] P. J. Denning *et al.*, "Computing as a discipline," *Commun. ACM*, vol. 32, no. 1, pp. 9–23, 1989.

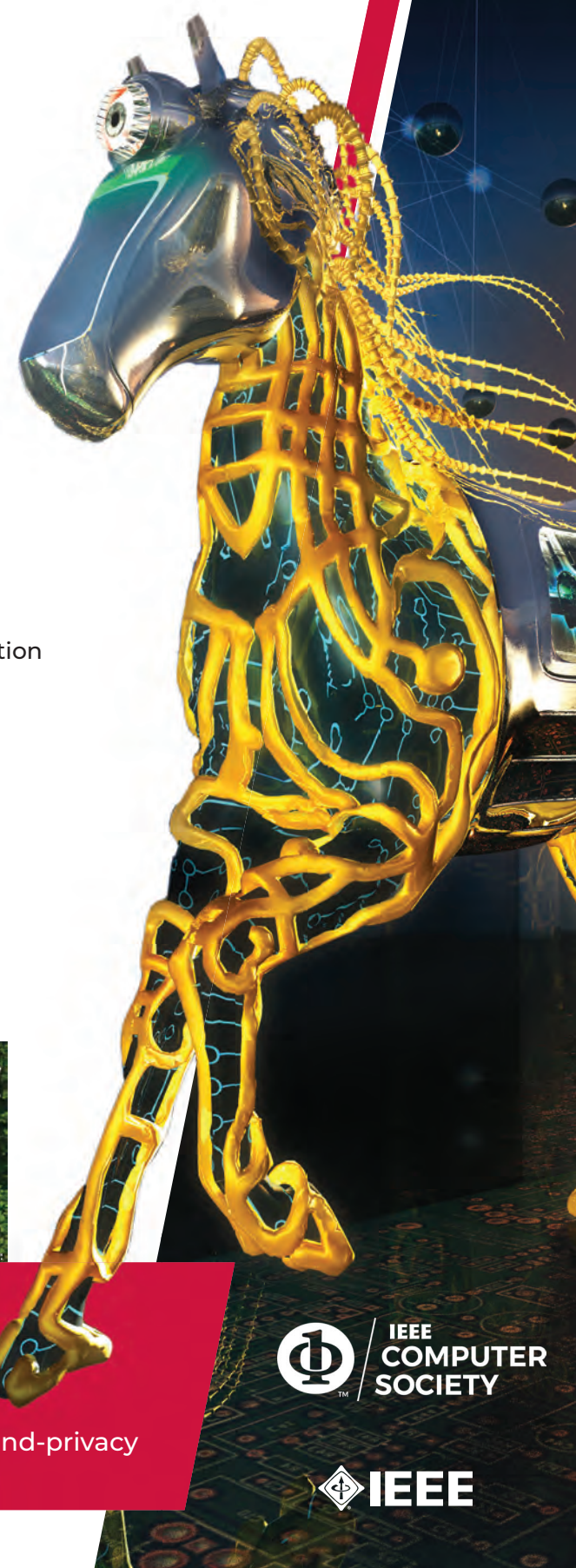
[4] H. Dreyfus, *What Computers Still Can't Do*. Cambridge, MA, USA: MIT Press, 1992. Originally published as "What computers can't do" in 1972.

[5] D. Hilbert and W. Ackermann, *Principles of Mathematical Logic*, 2nd translated ed. Providence, RI, USA: AMS Chelsea Pub., 1958, p. 122.

[6] M. Y. Vardi, "A logical revolution (slides for keynote)," in *Proc. 9th Joint Meeting Found. Softw. Eng.*, 2013, Art. no. 1.

IEEE

SECURITY & PRIVACY



***IEEE Security & Privacy* is a bimonthly magazine communicating advances in security, privacy, and dependability** in a way that is useful to a broad section of the professional community.

The magazine provides articles with both a practical and research bent by the top thinkers in the field of security and privacy, along with case studies, surveys, tutorials, columns, and in-depth interviews. Topics include:

- Internet, software, hardware, and systems security
- Legal and ethical issues and privacy concerns
- Privacy-enhancing technologies
- Data analytics for security and privacy
- Usable security
- Integrated security design methods
- Security of critical infrastructures
- Pedagogical and curricular issues in security education
- Security issues in wireless and mobile networks
- Real-world cryptography
- Emerging technologies, operational resilience, and edge computing
- Cybercrime and forensics, and much more

www.computer.org/security

Join the IEEE Computer Society for subscription discounts today!

www.computer.org/product/magazines/security-and-privacy

IEEE COMPUTER SOCIETY

IEEE

SUBMIT TODAY

IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING

SCOPE

The *IEEE Transactions on Sustainable Computing* (*T-SUSC*) is a peer-reviewed journal devoted to publishing high-quality papers that explore the different aspects of sustainable computing. The notion of sustainability is one of the core areas in computing today and can cover a wide range of problem domains and technologies ranging from software to hardware designs to application domains. Sustainability (e.g., energy efficiency, natural resources preservation, using multiple energy sources) is needed in computing devices and infrastructure and has grown to be a major limitation to usability and performance.

Contributions to *T-SUSC* must address sustainability problems in different computing and information processing environments and technologies, and at different levels of the computational process. These problems can be related to information processing, integration, utilization, aggregation, and generation. Solutions for these problems can call upon a wide range of algorithmic and computational frameworks, such as optimization, machine learning, dynamical systems, prediction and control, decision support systems, meta-heuristics, and game-theory to name a few.

T-SUSC covers pure research and applications within novel scope related to sustainable computing, such as computational devices, storage organization, data transfer, software and information processing, and efficient algorithmic information distribution/processing. Articles dealing with hardware/software implementations, new architectures, modeling and simulation, mathematical models and designs that target sustainable computing problems are encouraged.

SUBSCRIBE AND SUBMIT

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

www.computer.org/tsusc

IEEE COMPUTER SOCIETY

IEEE ComSoc IEEE Communications Society

CEDA IEEE Council on Electronic Design Automation

IEEE

IEEE TRANSACTIONS ON

BIG DATA

***IEEE Transactions on Big Data* is a quarterly journal that publishes peer-reviewed articles with big data as the main focus.**

The articles provide cross-disciplinary, innovative research ideas and applications results for big data including novel theory, algorithms, and applications. Research areas include:

- Big data
  - Analytics
  - Curation and management
  - Infrastructure
  - Performance analyses
  - Semantics
  - Standards
  - Visualization
- Intelligence and scientific discovery from big data
- Security, privacy, and legal issues specific to big data

Applications of big data in the fields of endeavor where massive data is generated are of particular interest.

www.computer.org/tbd

Join the IEEE Computer Society for subscription discounts today!

www.computer.org/product/journals/tbd

IEEE COMPUTER SOCIETY

IEEE

Keep up with the latest IEEE Computer Society publications and activities wherever you are.

Follow us:

- @ComputerSociety
- facebook.com/IEEEComputerSociety
- IEEE Computer Society
- youtube.com/ieeecomputersociety
- instagram.com/ieee_computer_society

IEEE COMPUTER SOCIETY

IEEE

IEEE TRANSACTIONS ON

COMPUTERS

Call for Papers: *IEEE Transactions on Computers*

Publish your work in the IEEE Computer Society's flagship journal, *IEEE Transactions on Computers* (*TC*). *TC* is a monthly publication with a wide distribution to researchers, industry professionals, and educators in the computing field.

TC seeks original research contributions on areas of current computing interest, including the following topics:

- Computer architecture
- Software systems
- Mobile and embedded systems
- Security and reliability
- Machine learning
- Quantum computing

All accepted manuscripts are automatically considered for the monthly featured paper and annual Best Paper Award.

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

IEEE COMPUTER SOCIETY

IEEE

SHARE AND MANAGE YOUR RESEARCH DATA

IEEE DataPort is an accessible online platform that enables researchers to easily share, access, and manage datasets in one trusted location. The platform accepts all types of datasets, up to 2TB, and dataset uploads are currently free of charge.

IEEE DataPort™

UPLOAD DATASETS AT IEEE-DATAPORT.ORG

IEEE

COMPUTER ARCHITECTURE LETTERS

***IEEE Computer Architecture Letters* is a forum for fast publication of new, high-quality ideas in the form of short, critically refereed technical papers.** Submissions are accepted on a continuing basis and letters will be published shortly after acceptance in IEEE Xplore and in the Computer Society Digital Library.

Submissions are welcomed on any topic in computer architecture, especially:

- Microprocessor and multiprocessor systems
- Microarchitecture and ILP processors
- Workload characterization
- Performance evaluation and simulation techniques
- Interactions with compilers and operating systems
- Interconnection network architectures
- Memory and cache systems
- Power and thermal issues at the architectural level
- I/O architectures and techniques
- Independent validation of previously published results
- Analysis of unsuccessful techniques
- Domain-specific processor architecture (embedded, graphics, network)
- High-availability architectures
- Reconfigurable computer architectures

www.computer.org/cal

Join the IEEE Computer Society for subscription discounts today!

www.computer.org/product/journals/cal

IEEE COMPUTER SOCIETY

IEEE

Conference Calendar

IEEE Computer Society conferences are valuable forums for learning on broad and dynamically shifting topics from within the computing profession. With over 200 conferences featuring leading experts and thought leaders, we have an event that is right for you. Questions? Contact conferences@computer.org.

SEPTEMBER

2 September

- VL/HCC (IEEE Symposium on Visual Languages and Human-Centric Computing), Liverpool, UK

15 September

- IISWC (IEEE Int'l Symposium on Workload Characterization), Vancouver, Canada
- QCE (IEEE Int'l Conf. on Quantum Computing and Eng.), Montreal, Canada

16 September

- ACSOS (IEEE Int'l Conf. on Autonomic Computing and Self-Organizing Systems), Aarhus, Denmark
- e-Science (IEEE Int'l Conf. on e-Science), Osaka, Japan

23 September

- MASS (IEEE Int'l Conf. on Mobile Ad-Hoc and Smart Systems), Seoul, South Korea

24 September

- CLUSTER (IEEE Int'l Conf. on Cluster Computing), Kobe, Japan
- IC2E (2024 IEEE Int'l Conf. on Cloud Eng.), Paphos, Cyprus

OCTOBER

6 October

- ICSME (IEEE Int'l Conf. on Software Maintenance and Evolution), Flagstaff, USA
- VISSOFT (IEEE Working Conf. on Software Visualization), Flagstaff, USA

7 October

- SCAM (IEEE Int'l Conf. on Source Code Analysis and Manipulation), Flagstaff, USA
- SecDev (IEEE Secure Development Conf.), Pittsburgh, USA

8 October

- DFT (IEEE Int'l Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems), Didcot, United Kingdom
- LCN (IEEE Conf. on Local Computer Networks), Normandy, France

10 October

- ICPADS (IEEE Int'l Conf. on Parallel and Distributed Systems), Belgrade, Serbia

11 October

- ICEBE (IEEE Int'l Conf. on e-Business Eng.), Shanghai, China

13 October

- LDAV (IEEE Symposium on Large Data Analysis and Visualization), St. Pete Beach, USA
- TopoInVis (IEEE Topological Data Analysis and Visualization), St. Pete Beach, USA
- VIS (IEEE Visualization and Visual Analytics), St. Pete Beach, USA

14 October

- VDS (IEEE Visualization in Data Science), St. Pete Beach, USA

21 October

- ISMAR (IEEE Int'l Symposium on Mixed and Augmented Reality), Bellevue, Washington, USA

27 October

- FOCS (IEEE Annual Symposium on Foundations of Computer Science), Chicago, USA

28 October

- CIC (IEEE Int'l Conf. on Collaboration and Internet Computing), Washington, DC, USA
- CogMI (IEEE Int'l Conf. on Cognitive Machine Intelligence), Washington, DC, USA
- ICNP (IEEE Int'l Conf. on Network Protocols), Charleroi, Belgium
- ISSRE (IEEE Int'l Symposium on Software Reliability Eng.), Tsukuba, Japan
- TPS-ISA (IEEE Int'l Conf. on Trust, Privacy and Security in Intelligent Systems, and Applications), Washington, DC, USA

30 October

- BDCloud (IEEE Int'l Conf. on Big Data and Cloud Computing), Kaifeng, China
- ISPA (IEEE Int'l Symposium on Parallel and Distributed

 2469-7087/24

Processing with Applications), Kaifeng, China
- SocialCom (IEEE Int'l Conf. on Social Computing and Networking), Kaifeng, China
- SpaCCS (IEEE Int'l Conf. on Security, Privacy, Anonymity in Computation and Communication and Storage), Kaifeng, China
- SustainCom (IEEE Int'l Conf. on Sustainable Computing and Communications), Kaifeng, China

NOVEMBER

3 November
- ICCD (IEEE Int'l Conf. on Computer Design), Milan, Italy

5 November
- CBDCom (IEEE Conf. on Cloud and Big Data Computing), Boracay Island, Philippines
- CyberSciTech (IEEE Cyber Science and Technology Congress), Boracay Island, Philippines
- DASC (IEEE Conf. on Dependable, Autonomic and Secure Computing), Boracay Island, Philippines
- PICom (IEEE Conf. on Pervasive and Intelligent Computing), Boracay Island, Philippines

13 November
- PRDC (IEEE Pacific Rim Int'l Symposium on Dependable Computing), Osaka, Japan

14 November
- SmartIoT (IEEE Int'l Conf. on Smart Internet of Things), Shenzhen, China

15 November
- MedAI (IEEE Int'l Conf. on Medical Artificial Intelligence), Chongqing, China

22 November
- IPCCC (IEEE Int'l Performance, Computing, and Communications Conf.), Orlando, USA

27 November
- BIBE (IEEE Int'l Conf. on Bioinformatics and Bioengineering), Kragujevac, Serbia

DECEMBER

4 December
- ICA (IEEE Int'l Conf. on Agents), Wollongong, Australia

9 December
- ICDM (IEEE Int'l Conf. on Data Mining), Abu Dhabi, United Arab Emirate

10 December
- RTSS (IEEE Real-Time Systems Symposium), York, UK

11 December
- ICKG (IEEE Int'l Conf. on Knowledge Graph), Abu Dhabi, United Arab Emirates
- ISM (IEEE Int'l Symposium on Multimedia), Tokyo, Japan

13 December
- DependSys (IEEE Int'l Conf. on Dependability in Sensor, Cloud & Big Data Systems & Applications), Wuhan, China
- DIKW (IEEE Int'l Conf. on Data, Information, Knowledge and Wisdom), Wuhan, China
- DSS (IEEE Int'l Conf. on Data Science and Systems), Wuhan, China
- HPCC (IEEE Int'l Conf. on High Performance Computing and Communications), Wuhan, China
- ICESS (IEEE Int'l Conf. on Embedded Software and Systems), Wuhan, China
- SmartCity (IEEE Int'l Conf. on Smart City), Wuhan, China

15 December
- BigData (IEEE Int'l Conf. on Big Data), Washington, District of Columbia, USA

18 December
- HiPC (IEEE Int'l Conf. on High Performance Computing, Data, and Analytics), Bangalore, India

19 December
- ESAI (Int'l Conf. on Embedded Systems and Artificial Intelligence), Fez, Morocco

IEEE COMPUTER SOCIETY CAREER CENTER

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.

IEEE COMPUTER SOCIETY

IEEE