Looking for the BEST Tech Job for You?

Come to the Computer Society Jobs Board to meet the best employers in the industry—Apple, Google, Intel, NSA, Cisco, US Army Research, Oracle, Juniper...

Take advantage of the special resources for job seekers—job alerts, career advice, webinars, templates, and resumes viewed by top employers.

www.computer.org/jobs
10  The Cyber-Physical Systems Revolution

14  A 21st Century Cyber-Physical Systems Education

29  Individualizing Cybersecurity Lab Exercises with Labtainers
Computer Science Education in 2018

Cyber-Physical Systems

10 The Cyber-Physical Systems Revolution
DIMITRIOS SERPANOS

14 A 21st Century Cyber-Physical Systems Education
JOHN A. STANKOVIC, JAMES W. STURGES, AND JON EISENBERG

Automotive Innovations

18 Improving the State of Automotive Software Engineering
ALIREZA HAGHIGHATKHAH, MARKKU OIVO, AHMAD BANIJAMALI, AND PASI KUVAJA

23 How Software Is Changing the Automotive Landscape
HANS AERTS AND HAN SCHAMINÉE

Education

29 Individualizing Cybersecurity Lab Exercises with Labtainers
MICHAEL F. THOMPSON AND CYNTHIA E. IRVINE

34 Computer Science Education in 2018
JEFFREY VOAS, RICK KUHN, CELIA PAULSEN, AND KIM SCHAFFER

Smart Cities

40 Bridging the Adoption Gap for Smart City Technologies: An Interview with Rob Kitchin
KATJA SCHECHTNER

Cloud Computing

44 Context-Aware Ubiquitous Biometrics in Edge of Military Things
ANIELLO CASTIGLIONE, KIM-KWANG RAYMOND CHOO, MICHELE NAPPI, AND STEFANO RICCIARDI

50 Keep It Simple: Bidding for Servers in Today’s Cloud Platforms
PRATEEK SHARMA, DAVID IRWIN, AND PRASHANT SHENOY

Departments

4 Magazine Roundup
8 Editor’s Note: An Even More Connected Future

Subscribe to ComputingEdge for free at www.computer.org/computingedge.
The IEEE Computer Society’s lineup of 13 peer-reviewed technical magazines covers cutting-edge topics ranging from software design and computer graphics to Internet computing and security, from scientific applications and machine intelligence to cloud migration and microchip design. Here are highlights from recent issues.

**Computer**

**Monitoring Eating Behaviors for a Nutritionist E-Assistant Using Crowdsourcing**

Researchers from the Ensenada Center for Scientific Research and Higher Education and the Sonora Institute of Technology have created Lucy, a digital assistant that monitors eating behaviors to help users lose weight. The e-assistant’s design was informed by a study of clients in a nutrition clinic, as well as by crowdsourcing, to evaluate six approaches to assessing nutritional content or caloric intake based on meal photos. Learn more about Lucy in this article from the March 2018 issue of *Computer*.

**Computing in Science & Engineering**

**Pythran: Crossing the Python Frontier**

Use of the Python language in scientific computing has always been characterized by the co-existence of interpreted Python code and compiled native code, written in languages like C or Fortran. In this article from the
March/April 2018 issue of Computing in Science & Engineering, the author takes a fresh look at the problem and introduces Pythran, a new optimization tool designed to efficiently handle unmodified Python code.

IEEE Annals of the History of Computing

Anne-Louise Guichard Radimsky: An Educator and a Champion for Diversity in Computing

The life and career of Anne-Louise Guichard Radimsky intersects continents, cultures, and disciplines, and reveals the pathways available to women in the early years of computer science in the US and France. After accepting a scholarship to study computer science in the US in 1966, Anne-Louise Guichard (later Radimsky) embarked on a lifelong journey to becoming a cherished computer science educator and mentor. Learn more about Anne-Louise and her work in the October–December 2017 issue of IEEE Annals of the History of Computing.

IEEE Cloud Computing

Blockchain: A Panacea for Healthcare Cloud-Based Data Security and Privacy?

One particular trend observed in healthcare is the progressive shift of data and services to the cloud, partly due to convenience (for example, the availability of complete patient medical history in real time) and savings (for example, the economics of healthcare data management). There are, however, limitations to using conventional cryptographic primitives and access-control models to address security and privacy concerns in an increasingly cloud-based environment. In this article from the January/February 2018 issue of IEEE Cloud Computing, researchers study the potential of using blockchain technology to protect healthcare data hosted within the cloud.

IEEE Computer Graphics and Applications

ColorMoves: Real-time Interactive Colormap Construction for Scientific Visualization

In this article from the January/February 2018 issue of IEEE Computer Graphics and Applications, researchers present ColorMoves, an interactive tool that promotes exploration of scientific data through artist-driven color methods in a unique and transformative way. They also discuss the power of contrast in scientific visualization, the design of the ColorMoves tool, and the tools application in several science domains.

IEEE Intelligent Systems

Performance Estimation and Dimensioning of Team Size for Multirobot Patrol

The performance of multirobot patrolling teams heavily relies on the number of cooperative mobile robots that carry out the patrol mission. In the deployment phase, such systems are typically dimensioned to perform a certain way. However, this is often done empirically using trial and error approaches. This article from the November/December 2017 issue of IEEE Intelligent Systems investigates the problem of estimating the performance of teams of robots in patrol missions, with the ultimate goal of providing the appropriate number of robots before the start of the mission.

IEEE Internet Computing

Delay-Tolerant Networking for Long-Term Animal Tracking

Enabling Internet connectivity for mobile objects that do not have a permanent home or regular movements is a challenge due to their varying energy budget, intermittent wireless connectivity, and inaccessibility. In this article from the January/February 2018 issue of IEEE Internet Computing, researchers present a framework that offers robust data collection, adaptive execution of sensing tasks, and flexible remote configuration of devices deployed on nomadic mobile objects such as animals. They present the main challenges they encountered, the design of software building blocks that address these challenges, and examples of the data they collected on flying foxes (fruit bats).

IEEE Micro

Two Billion Devices and Counting

Mobile computing has grown drastically over the past decade. Despite the rapid pace of advancements,
mobile device understanding, benchmarking, and evaluation are still in their infancies, both in industry and academia. In this article from the January/February 2018 issue of *IEEE Micro*, Google researchers present an industry perspective on the challenges facing mobile computer architecture—specifically involving mobile workloads, benchmarking, and experimental methodology—with the hope of fostering new research within the community to address pending problems. These challenges pose a threat to the systematic development of future mobile systems, which, if addressed, can elevate the entire mobile ecosystem to the next level.

**IEEE MultiMedia**

*An NFV-Based Video Quality Assessment Method over 5G Small Cell Networks*  
In this article from the October–December 2017 issue of *IEEE MultiMedia*, researchers discuss a video-quality assessment mechanism for next-generation (5G) mobile networks that uses small cell deployment architecture to implement a virtual network function to enable in-service monitoring of delivered video quality.

**IEEE Pervasive Computing**

*Magic Room: A Smart Space for Children with Neurodevelopmental Disorder*  
The Magic Room is a smart space for use by children with neurodevelopmental disorder (NDD) and their caregivers, designed in cooperation with NDD specialists and currently deployed at two therapeutic centers in Italy. It supports multimodal embodied interaction by providing controllable stimuli to the vestibular, proprioceptive, and tactile sensory systems through ambient sound and visual projections, soap bubbles, aromas, lights, toys, and other physical objects. Learn more about the encouraging results of an exploratory study involving the Magic Room in the January–March 2018 issue of *IEEE Pervasive Computing*.

**IEEE Security & Privacy**

*House Rules: Designing the Scoring Algorithm for Cyber Grand Challenge*  
The key driving force behind any capture-the-flag competition is the scoring algorithm—the Cyber Grand Challenge (CGC) was no different. The scoring algorithm design for both the CGC Qualifier Event and the CGC Final Event focused heavily on encouraging development of automated reasoning about software and its inputs, discouraging collusion and cheating, and representing real-world constraints to ensure resulting solutions developed by CGC competitors were well-positioned for adoption outside the CGC. In this article from the March/April 2018 issue of *IEEE Security & Privacy*, researchers describe design considerations for the scoring algorithms, how these algorithms incentivized competitors to achieve their goals, and effects these decisions had on the resulting CGC gameplay.
IEEE Software

Continuous Delivery: Building Trust in a Large-Scale, Complex Government Organization

For many software development teams, the first aspects that come to mind regarding continuous delivery (CD) are the operational challenges and competitive benefits. However, the authors of this article from the March/April 2018 issue of IEEE Software discovered that CD was much more—it was a survival technique. The authors present how and why they applied CD in a large governmental project to create a collaborative development environment. They share the challenges they faced, the strategies they used to overcome them, and a set of lessons learned that can be valuable for readers.

IT Professional

The Role of a Customer Data Platform

In this article from the January/February 2018 issue of IT Professional, the author explains that by providing access to data from numerous systems in one database and supporting the systems that can produce an appropriate customer experience, a customer data platform overcomes the limitations imposed by fragmented point solutions and presents a holistic approach to customer interactions.

Computing Now

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

IEEE Annals of the History of Computing

From the analytical engine to the supercomputer, from Pascal to von Neumann, from punched cards to CD-ROMs—IEEE Annals of the History of Computing covers the breadth of computer history. The quarterly publication is an active center for the collection and dissemination of information on historical projects and organizations, oral history activities, and international conferences.

www.computer.org/annals
EDITOR’S NOTE

An Even More Connected Future

Cyber-physical systems (CPSs) are networks of physical and computational components that interact with one another, providing critical infrastructure and laying the groundwork for future smart cities. These systems are poised to bring advances in healthcare, traffic management, power generation, disaster and emergency response, and more. In this issue of ComputingEdge, two Computer articles provide insights into the CPS realm. In “The Cyber-Physical Systems Revolution,” the author goes over CPS basics and how this field will revolutionize economies and social processes, among other things. “A 21st Century Cyber-Physical Systems Education” explores how best to prepare graduates for a world in which CPSs are increasingly ubiquitous.

Another CPS facet is smart vehicles. The authors of IEEE Software’s “Improving the State of Automotive Software Engineering” study the existing literature on the subject of automotive software engineering and make practitioner-oriented recommendations for automakers to stay ahead of the game. In “How Software Is Changing the Automotive Landscape,” also from IEEE Software, the authors examine what’s happened in the automotive and navigation domains over the past few years and predict what will come in an even more connected future.

Education is key for any industry, but especially in those where the landscape is always changing, such as computing. In IEEE Security & Privacy’s “Individualizing Cybersecurity Lab Exercises with Labtainers,” the authors discuss the hurdles to creating and assessing exploratory cybersecurity laboratory exercises and three possible approaches to lab parameterization. In IT Professional’s “Computer Science Education in 2018,” six educators are asked three questions about the current state of computer science education, software engineering, and licensing software engineers using a “Body of Knowledge” approach.

The concept of smart cities (which will be made up of CPSs) has become increasingly popular, but it is difficult to bridge the gap between what city administrators want and what technology developers offer. In IEEE Pervasive Computing’s “Bridging the Adoption Gap for Smart City Technologies: An Interview with Rob Kitchin,” professor and European Research Council Advanced Investigator Rob Kitchin answers questions about the social, political, and economic implications of smart cities and how to create solutions that work for cities and their citizens alike.

Finally, cloud computing is an important element of CPS and future technologies. In IEEE Cloud Computing’s “Context-Aware Ubiquitous Biometrics in Edge of Military Things,” the authors examine cloud computing’s role in enabling user authentication and monitoring in military and battlefield applications. The authors of IEEE Internet Computing’s “Keep It Simple: Bidding for Servers in Today’s Cloud Platforms” present simple and effective bidding strategies for public cloud users and provide motivation for new research directions in cloud resource management and fault tolerance.
Recognizing Excellence in High Performance Computing

Nominations are Solicited for the

SEYMOUR CRAY
SIDNEY FERNBACH
& KEN KENNEDY AWARDS

SEYMOUR CRAY COMPUTER ENGINEERING AWARD
Established in late 1997 in memory of Seymour Cray, the Seymour Cray Award is awarded to recognize innovative contributions to high performance computing systems that best exemplify the creative spirit demonstrated by Seymour Cray. The award consists of a crystal memento and honorarium of US$10,000.

This award requires 3 endorsements.

Sponsored by: IEEE computer society

SIDNEY FERNBACH MEMORIAL AWARD
Established in 1992 by the Board of Governors of the IEEE Computer Society. It honors the memory of the late Dr. Sidney Fernbach, one of the pioneers on the development and application of high performance computers for the solution of large computational problems. The award, which consists of a certificate and a US$2,000 honorarium, is presented annually to an individual for “an outstanding contribution in the application of high performance computers using innovative approaches.”

This award requires 3 endorsements.

Sponsored by: IEEE computer society

ACM/IEEE-CS KEN KENNEDY AWARD
Established in 1992 by the Board of Governors of the IEEE Computer Society. It honors the memory of the late Dr. Sidney Fernbach, one of the pioneers on the development and application of high performance computers for the solution of large computational problems. The award, which consists of a certificate and a US$5,000 honorarium, is awarded jointly by the ACM and the IEEE Computer Society for outstanding contributions to programmability or productivity in high performance computing together with significant community service or mentoring contributions.

This award requires 2 endorsements.

Cosponsored by: ACM computer society

Deadline: 1 July 2018
All nomination details available at http://awards.computer.org
The Cyber-Physical Systems Revolution

Dimitrios Serpanos, ISI/ATHENA and University of Patras

Cyber-physical systems constitute a disruptive technology across many industries, with a strong impact on economies and social processes. Their applications in many domains, from manufacturing to agriculture and from critical infrastructure to assistive living, will challenge technology, business, law, and ethics.

Cyber-physical systems (CPS) integrate computational and physical components to implement a process in the real world. Since the beginning of computing, there have been efforts to automate complex processes through the inclusion of computing systems, as demonstrated by the well-known Apollo Guidance Computer in the 1960s, which was used for guidance, navigation, and control. The technological advances of the last decades on fronts such as sensors, instrumentation, networking, and embedded computing have enabled us to develop systems and applications that have changed our daily lives. Automatic building power management, automated insulin delivery pumps, self-braking cars, and surveillance drones are just a few examples of conventional CPS that affect everyday life in diverse application areas.

CPS constitute a disruptive technology, bringing innovation to many industries because of their potential to integrate technologies from various sectors, transform traditional processes in several application areas, and enable new processes. Consumers experience transformations in many of the services they receive through conventional devices, from home automation and energy management to health services and entertainment. Personalized healthcare is one of the first changes, especially as medical innovation increasingly offers personalized treatments to citizens, with the support of sophisticated health devices at home and in health centers and hospitals; examples of CPS health applications are systems that monitor patients continuously and deliver medication on the fly; systems that enable remote monitoring of patients; and systems that support the movement of individuals with disabilities by, for example, sensing and adjusting the actions of artificial limbs.

Energy systems are being transformed at all levels because of CPS. Consumers, individuals, and organizations are already experiencing significant advantages from
the exploitation of smart systems that manage energy consumption in increasingly smart buildings that sense power needs and enable delivery. Energy producers exploit CPS to identify the need for energy and adjust their plants to deliver the required energy where and when needed, saving significant resources. Electric power distribution operators increasingly employ CPS to monitor and manage their distribution systems in real time to avoid outages and appropriately satisfy their consumers’ needs.

The electric grid, already being transformed to a smart grid, constitutes part of the critical infrastructure of many countries, and is managed and controlled by CPS. Its continuous and reliable operation is necessary for these countries’ economies as well as for the well-being of their citizens. Its service disruptions can lead to significant problems in everyday life, influencing aspects such as safety, productivity, and health.

Water management, transportation, manufacturing, and communications also constitute significant components of critical infrastructure and of emerging smart cities. Smart traffic management, autonomous vehicles, automated water leakage detection and prevention systems, and self-healing networks are transforming critical infrastructures’ management and effectiveness. Several innovations are changing traditional processes in disruptive ways, for instance, allowing the personalization of mass-produced products in manufacturing—considered by many to be the fourth industrial revolution. Smart agriculture, enabled by CPS management of agricultural production, is another example of a CPS application with a strong, widescale impact. CPS are already having a visible impact on many fronts, including robotics, security, safety, and military.

CPS’s progress has enabled the emergence and fast growth of the Internet of Things (IoT). Exploiting Internet technology to transfer data over heterogeneous communication technologies, both wired and wireless, has provided opportunities to combine systems across distances in distributed applications and processes, collect data remotely for process analysis and optimization, and manage systems remotely and reliably at a low cost. IoT-connected CPS as “things,” combined with data analytics and artificial intelligence, boost development of a wide range of smart systems, environments, applications, and services, from smart cities and governance to smart manufacturing and control of critical infrastructure.

CHALLENGES
CPS differ from classical computational and information systems in many aspects. Their structure includes a wide range of heterogeneous technologies for computational components, networks, sensors, and actuators. Their interaction with the physical world raises issues about their robustness and resilience as well as about liability when CPS actions violate expectations or even regulations in applications, for example, in health or transport systems. Furthermore, their adoption for automating processes in domains ranging from manufacturing to agriculture is expected to change business models by taking humans out of the loop in many services, which will have major societal consequences. Thus, CPS’s projected impact across industries and applications will create significant technical, legal, and societal challenges.

Considering the applications of CPS and their related technologies—computational, network, and instrumentation—we must address these challenges. CPS must be architected, designed, and implemented to be easily extendable and scalable. As systems are becoming increasingly interconnected, CPS must extend their functionality to integrate heterogeneous communication technologies and become part of larger systems that implement even more sophisticated processes. The design, operation, and management of such complex systems requires the ability to describe them at various levels of abstraction, enabling analyses and decisions for technical, operational, management, and business purposes. The need for scalability, modularity, and composability leads to significant technical issues in the area of systems-of-systems, where correctness and system verification are challenging problems, especially in light of applications with timing constraints and continuous operation requirements. An important challenge in this direction is the integration of traditional IT systems with operational technology (OT) systems. OT systems differ from traditional IT systems in many ways, from purpose, computing components, communication technologies, and interfaces to ownership and management. The integration of these technologies poses
The security challenges go beyond computations and communication in CPS. They also include reliable interaction with the physical world, especially when considering that the physical environment might “behave” in unexpected and unpredictable ways; independently, CPS have to adjust and operate reliably in a well-defined and predictable fashion with safety as a major priority, especially in environments where hazardous conditions might occur, such as power systems, autonomous vehicles, and health systems. Security is a prerequisite for safety because security breaches might lead to safety hazards—although not necessarily.

Another important aspect of security relates to the privacy of users. Security attacks on home devices, for example, can lead to sensitive information leakage. Moreover, the expected collection of (even anonymous) data from CPS operation over long time intervals might disclose behavioral patterns of users that enable their identification and characterization beyond the scope of applications, violating privacy rights.

Reliability and dependability are a necessity in CPS, especially in environments where continuous operation is required, as in the domains of critical infrastructure, manufacturing, health, autonomous vehicles, and so on. There is a need to address these properties combined with security, because malicious attacks often attempt to disrupt the continuous operation of the systems. Attacks on power systems, such as the recent attack on the Ukrainian power grid, have demonstrated the urgency of addressing these aspects in a unified way.

The integration of CPS across domains raises the challenge of interoperability. Data interoperability has been already identified as a major priority, because data have to be interpreted consistently within processes at all levels of the overall distributed CPS. However, data interoperability is only one aspect of the interoperability challenge, which needs to address computational aspects in addition to data. Composing complex CPS from components that already implement simpler processes requires consistent interpretation of the context and the semantics of operations, commands, and subprocesses, setting the foundation for machine interpretation, inferencing, logic, and knowledge discovery in order to enable seamless integration and compose a correct and resilient overall process.

The distributed structure of CPS also raises timing challenges. An integral part of many CPS applications is meeting real-time requirements. Many of the manufacturing processes, for example, require periodic actions and mandate time criticality; many monitoring systems detect conditions that need a reaction within a specific time interval. In many cases, violation of a deadline leads to unsafe states and results in undesirable conditions such as accidents. Combining systems with different clocks that operate at different granularities, and have different real-time requirements in terms of scale and priorities is a well-known challenge in computing and requires innovative methods to enable a unified timing scheme for the overall system.

Employment of CPS and their effect on emerging applications raises legal and societal challenges. Clearly, privacy concerns lead to legal actions and challenge lawmakers to establish legal frameworks for the appropriate storage, processing and use of sensitive data. Although there are significant efforts in progress, such as the General Data Protection Regulation (GDPR) in the EU, the unanticipated usage of collected data as well as the unexpected breaches of privacy and leakage of sensitive information pose a continuous challenge to evolve legal frameworks that protect consumers and organizations effectively.

Digital rights management is another aspect of CPS applications that has significant legal ramifications. Although the issue has been targeted for more than a decade at the online entertainment business, the pervasiveness of CPS makes the problem more acute, covering issues beyond media, including designs, software, and licenses on a large scale.

Additionally, despite efforts toward resilient CPS, unexpected circumstances are bound to occur, leading to violations of safety and accidents. Liability is a major issue in such cases—are human operators, manufacturers, or maintainers liable? What procedures are in place to investigate sources of problems and place liability appropriately? Differences in legal
systems across the globe, in conjunction with emerging distributed CPS processes across borders, present unique legal challenges.

The adoption of CPS to automate processes is increasingly taking humans out of control loops, but creating new opportunities in the maintenance and management of the control loops. Thus, policies are needed that will help citizens transition effectively into these new employment roles. At the same time, ethical questions are raised about limits: where, how, and when automated processes should be replacing human operation and judgement. This relates not only to processes far from immediate human oversight but also to systems immediately used by people for their health and well-being, such as assistive living and artificial prosthetic parts.

**INITIATIVES**

National initiatives and large consortia efforts have emerged to develop frameworks that will enable fast and effective growth addressing the emerging challenges. Most notable among them are the Industrie 4.0 initiative in Germany, the Industrial Internet in the US, and the Society 5.0 initiative in Japan.

Industrie 4.0 is a strategic initiative to enable Germany to maintain a leading role in manufacturing.\(^1\) The goal is efficient and low-cost production with flexible workflows. This will be achieved through widespread use of CPS in manufacturing and production processes, inserting intelligence in systems and processes, and coordinating them into more complex but flexible processes that produce high-quality products at low cost. The smart factory concept, which achieves the initiative’s goals, employs a hierarchy of CPS with smart machines interconnected to establish smart plants, which are then combined to establish smart factories. The concept targets flexibility, autonomy, resilience, safety, efficiency, and low cost, enabling all actors—customers, operators, and manufacturers—to monitor the system parameters of their interest.

The Industrial Internet of Things (IIoT) focuses on the application of IoT to the industrial sector, including not only the industrial production process but other processes such as asset management and maintenance. IIoT addresses efficient and effective operations and interoperability, taking into account emerging and future services as well as the stakeholders involved in the related devices, CPS, communications, service provision, and business development. The need for standards and reference architectures is addressed by several efforts, most notably the International Telecommunication Union (ITU) ITU-T Y.2060 recommendation\(^2\) and the Industrial Internet Consortium’s (IIC’s) Industrial Internet Reference Architecture.\(^3\) The ITU recommendation addresses IoT, in general, but includes applications for IIoT such as a smart grid.

Importantly, the ITU recommendation also presents business models for IoT. The IIC is a detailed reference architecture that can be considered an elaboration of the ITU reference architecture addressing important aspects for all categories of stakeholders. Importantly, NIST has been developing a CPS framework to enable fast development and deployment of CPS in the nation’s economy.\(^4\)

Society 5.0, also called Super Smart Society, is a Japanese initiative to integrate CPS with IoT, big data technologies, and AI into every industry and all aspects of society to address societal challenges.\(^5\) High-priority applications are in healthcare, transportation and mobility, infrastructure maintenance, and the financial sector.

CPS are revolutionizing economies and social processes. Their effective development, adoption, and use require breakthroughs on several fronts, including policy, law, business, and social sciences as well as technology.\(^\square\)

**REFERENCES**


**DIMITRIOS SERPANOS** is director of the Industrial Systems Institute/ATHENA and a professor of electrical and computer engineering at the University of Patras. He is a Senior Member of IEEE and a member of ACM, AAAS, and NYAS. Contact him at serpanos@computer.org.

**This article originally appeared in** Computer, vol. 51, no. 3, 2018.
A 21st Century Cyber-Physical Systems Education

John A. Stankovic, University of Virginia
James W. Sturges, Independent Consultant
Jon Eisenberg, National Academies of Sciences, Engineering, and Medicine

The rapid rate of change in the computing and engineering domains has our educational institutions on high alert. The authors explore how best to prepare graduates for a world in which cyber-physical systems are increasingly ubiquitous.

Using software to define capabilities in engineered systems provides extraordinary flexibility along with the promise of unprecedented growth in the economy, functionality, safety, and accuracy of control and operational decision-making. An exciting technological revolution is underway to engineer cyber-physical systems (CPS) that integrate computational and physical elements and manage the significant and intimate couplings between the two aspects. These complex systems increasingly operate in loosely supervised and complex environments, interact with the Internet and its services, operate with a high degree of autonomy, involve humans-in-the-loop, and, often, are safety critical. Such systems must address complications, such as systems-of-systems challenges and desired failure modes, if they are to achieve the desired levels of safety, security, and privacy. In this era of “smart things,” virtually all industries are rapidly implementing CPS.

Interdisciplinary skillsets to invent, design, build, and deploy these systems are more necessary than ever before. The NSF recently sponsored a multiyear study by the National Academies of Sciences, Engineering, and Medicine to develop clearly articulated criteria for curricula to address effective CPS education.¹ The study found that all computing and engineering fields will make widespread use of CPS, and the workforce must have access to domain experts knowledgeable in CPS principles.

CPS AS AN ENGINEERING AND COMPUTING DOMAIN

In the course of the study, experts in a wide array of industrial sectors—including agriculture, transportation, and medical devices—reported the changing nature of their products and services as well as the corresponding
challenges they face in recruiting engineers with the required skills. Thus, multiple paths to teaching CPS knowledge are required to meet labor demands. For example, CPS survey courses taught at the undergraduate level will provide students with a basic understanding of such systems and the key challenges to their design, which will be necessary in any engineering discipline, including aerospace, civil, or mechanical. In addition, engineering programs that include a CPS concentration or focus would accordingly provide a stronger, more deliberate foundation for CPS work. In addition, we posit the need for a new type of engineer—a CPS engineer—along with a corresponding bachelor’s-level CPS engineering degree to create a cadre of engineers well versed in both the cyber and physical issues to meet growing industry needs for this expertise.

Although a handful of master’s-level programs exist, they focus on embedded systems or CPS, with a chiefly electrical engineering or computer science slant. An MSc program in CPS for graduates of other engineering fields, such as mechanical or civil engineering, would also be valuable. If CPS follows the lead of other engineering disciplines, PhD programs will help educate tomorrow’s CPS faculty, and PhD-level engineers will fill important technical leadership roles in industry.

ESTABLISHING A CPS CURRICULUM

Depending on the particulars of each university or college, it’s likely that a variety of approaches will be tried, reflecting existing department structures and curricula, faculty expertise, and available resources. Designing a CPS course or degree program is quite complex and involves, for example, a careful balancing of physical and cyber aspects and general CPS and application knowledge. Although CPS degree curricula are in their infancy, they’ll evolve substantially as CPS classes and systems are more widely deployed. Moreover, like most engineering degree programs, those in CPS will face the challenge of prioritizing topics to fit in a manageable four-year program of study. To help guide those developing CPS curricula, the report includes model curricula from multiple perspectives.

Given that the potential content for CPS programs is quite broad and rapidly evolving, the report focuses on principles and intellectual foundations rather than an array of specific techniques or facts. It identifies six overarching foundations for a CPS curriculum, as described in the following sections.

Foundation 1: Basic computing concepts
Expertise in CPS can’t be achieved through only one or two programming classes, as it requires solid training in computing. The basic computing concepts listed below should be taught using case studies and examples from the physical domain. These concepts include embedded hardware; data structures and algorithms; models of computation, including automata theory (relevant to the finite state machines widely used in CPS) and discrete event systems; programming; software engineering and model-based design; real-time operating systems; and programming for networks.

Foundation 2: Computing for the physical world
There’s a need for computing foundations to embrace physical-world properties and constraints. Real-world complexities often give rise to situations neither anticipated by the system designers nor addressed by the software, and thus often result in failure. System and software design and implementation must take into account the resource limitations of the platforms themselves, as well as conditions that the real world imposes on the platform. Students will need to thoroughly understand the following concepts: properties of sensors and analysis of signals; programming with sensors and actuators in open environments and with multiple modalities; real-time embedded systems; resource management and constraints such as time, memory size, and power; and techniques such as redundancy and fault-tolerance for managing unreliability in physical systems.

Foundation 3: Discrete and continuous mathematics
Both discrete and continuous mathematics are foundational skills for all CPS engineers. CPS deals with both continuous and discrete systems; thus it is critical for students to learn how to deal with that integration. Concepts students will need to understand include graph theory and combinatorics; probability, statistics, and stochastic processes; logic; linear algebra; and calculus and differential equations.
Foundation 4: Cross-cutting application of sensing, actuation, control, communication, and computing

This foundation is essential due to the cross-cutting nature of CPS, as well as control over communication networks and sensing, signal processing, and actuation with real-time constraints. The interdisciplinary nature of the topic must be intrinsic to all aspects of the curriculum. Knowledge of control, signal processing, and embedded software design and implementation are at the core of this foundational principle. To ensure adequate coverage of this concept, curricula will cover

- Control principles including linear and nonlinear systems, stochastic systems, adaptive control, system identification, and hybrid control;
- Optimization and optimal control of dynamic systems;
- Networking concepts including wireless communications, synchronous and asynchronous communications, and ad hoc networking;
- Real-time analysis including task models describing real-world information sources, time-triggered or event-triggered control, and decision-making with noisy data;
- Signal processing using control, computation, and communication models;
- Safety, reliability, and dependability;
- Security and privacy;
- Impact of physical properties on software requirements;

Foundation 5: Modeling heterogeneous and dynamic systems and integrating control, computing, and communication

CPS modeling requires a complete picture of control, communications, and computing—with emphasis on representing and accounting for modularity, abstraction, uncertainty, and heterogeneity. Relevant techniques include linear and nonlinear models, stochastic models, and discrete-event and hybrid models, and associated design methodologies based on optimization, probability theory, and dynamic programming are needed. Key concepts of this foundation include properties of the physical world, including uncertainty and risk; properties of computational devices, including computational and power limits; properties of communication systems, including limitations of wireless communications; error detection and correction; merging physical and computational modeling; and commonalities between signals and systems and finite-state automata.

Foundation 6: CPS system development

CPS development, from determining initial requirements to certification, requires a lifecycle view that parallels traditional systems engineering.

- Human factors related to humans-in-the-loop as well as behavioral aspects; and
- Networked control.

Beyond these intellectual foundations, the report brings to light highlights several other important elements of a CPS curriculum. Successful development of CPS requires attention to system characteristics, including security and privacy, interoperability, reliability and dependability, power and energy management, safety, stability and performance of dynamic and stochastic systems, as well as human factors and usability. The study observes that, in keeping with the best practices in engineering, these topics are best introduced early and infused throughout CPS coursework and projects.

Broader trends in engineering education, including the observation that rapid change and preparation for continual reeducation is particularly important for an emerging and rapidly changing area like CPS, are also covered in the report. Likewise, the inherently interdisciplinary nature of CPS and the growing complexity and scale of engineered systems place a premium on those able to work well in teams and communicate effectively with both technical and public audiences.

O f course, no discussion of curriculum development is complete without some discussion of the challenges inherent in tackling a new field of engineering. The report emphasizes building awareness of CPS opportunities among students in K-12 and incoming college freshmen. In
addition, it describes how educational institutions must invest to develop, recruit, and retain the faculty needed to provide an up-to-date CPS education. Add to this the need for new instructional materials, laboratory facilities, and testbeds to effectively support CPS courses and programs. In addition to federal agencies that support STEM education, contributions from industry, professional societies, and colleges and universities can all play important roles in building such resources and capabilities.

Success in these endeavors will have significant payoffs. An engineering workforce with high proficiency in CPS skills will help realize the full potential to engineer increasingly capable, adaptable, and trustworthy systems.

ACKNOWLEDGMENTS
This work was supported by the National Science Foundation under Grant No. CNS-1341078.

REFERENCES

This article originally appeared in Computer, vol. 50, no. 12, 2017.

JOHN A. STANKOVIC is the BP America Professor at University of Virginia. Contact him at stankovic@cs.virginia.edu.

JAMES STURGES retired from Lockheed Martin Corp. as Director of Mission Success. Contact him at jwsturges@gmail.com.

JON EISENBERG is the director of the Computer Science and Telecommunications Board of the National Academies of Sciences, Engineering, and Medicine. Contact him at jeisenbe@nas.edu.
Improving the State of Automotive Software Engineering

Alireza Haghighatkhah, Markku Oivo, Ahmad Banijamali, and Pasi Kuvaja

THE AUTOMOTIVE INDUSTRY is fundamentally changing by becoming software intensive, rather than mechanically intensive. Now, innovation and competitiveness rely principally on software engineering competence. You could argue that today’s premium vehicles run on code as much as on fuel.

Software and electronics in vehicles have led to innovative functionalities; reduced fuel consumption; and improved performance, comfort, and safety. Moreover, software has negligible replication costs and enables reuse, mass differentiation, and customization. Thus, software engineering gives automakers a competitive advantage.

To stay ahead of the game, industry players must continuously improve software engineering technologies, as well as their underlying processes and practices. The literature contains much knowledge that could help automakers improve industry practices. Via an extensive systematic mapping study, we aggregated and synthesized information from academic and practitioner-oriented literature, analyzed the material, and developed a set of practitioner-oriented recommendations.

**Literature Review**

To find studies on automotive software engineering (ASE) relevant to our research, we searched five major scientific databases and the Society of Automotive Engineers’ digital library of practitioner-oriented articles. From 4,348 retrieved records, we found 679 articles relevant to our scope, including 247 academic studies, 240 industry–academic collaborative works, and 192 practitioner-oriented articles. (The full list of analyzed studies and data extraction results is at www.dropbox.com/s/n7ix7h5yk9puavs/ASE_SMS_Repository_17022016.xlsx?dl=0). We then examined these articles carefully.

**The Automotive Domain’s Characteristics**

While the automotive industry shares features with other domains, it has its own characteristics, such as:

- heterogeneous and distributed software,
- a distributed development environment,
- many automobile variants and configurations,
- cost pressure and the predominance of unit-based cost models, and
- the prevalence of legacy systems.

These characteristics yield several fundamental ASE challenges and affect the automotive industry in
general, in areas such as long-term
countries.

What Do We Know about ASE?
Figure 1 shows topics identified in the ASE literature and the number of articles that mentioned them, thereby revealing the main areas of concern. Three topics—system or software architecture and design, system or software qualification testing, and software reuse—were addressed most often.

AUTOSAR
To manage automotive-software
development’s complexity, the industry has developed and adopted several standards. For example, AUTOSAR (Automotive Open System Architecture) creates an open and standardized software architecture for vehicular electronic control units. Surveys show that AUTOSAR yields advantages such as standardization, efficient development, shorter development lead time with the ability to introduce improvements and new features faster, software reuse, and interoperability. It also has drawbacks such as complexity; the initial investment in new processes, practices and tools; and a steep learning curve.

Automakers will have to migrate their existing legacy systems to an AUTOSAR-compliant architecture to avoid experiencing interoperability and compatibility problems, as well as hindering innovation.

Testing
An empirical automotive-software-
testing study reported 26 challenges and 15 solutions. The challenges primarily entail requirements, test management, and automation. The solutions involve requirements management, competence management, quality assurance and standards, test automation and tools, agile incorporation, and test management.

A particular problem deals with the ability to exchange test specifications among teams and systems. In this situation, the use of different test languages—which have various syntaxes, semantics, data formats, and interface descriptions—causes difficulties. To avoid this problem, practitioners must adopt formal, test-platform-independent protocols.

Existing evidence suggests that automatically and manually derived model-based test suites are much better at detecting requirement errors than handcrafted test suites. Findings show that automated test execution is heavily used but automated test generation isn’t because current tools don’t support it.

Given the enormous range of products and variants, exhaustive testing of entire automotive software product lines (SPLs) is unrealistic. Several proposed solutions would derive small representative sets of test cases that could demonstrate the entire product line’s correctness.

In addition, continuous-controller testing is demanding because of its many inputs and configuration parameters. To deal with this, researchers have proposed several search-based testing approaches, which practitioners could use to automate complex-system verification.

Requirements Engineering
Studies have found that requirements engineering (RE) is often inadequate in practice and that its processes aren’t well defined. Challenges
include the lack of a comprehensive requirements model, the unsystematic capture of requirements, the need to model a large number of requirements, the modeling of nonfunctional characteristics, requirements volatility, and inadequate tool support.1

Natural-language requirements specification, although still common, is prone to problems for several reasons. Although structured natural-language specification affords some advantages, it doesn’t support automated analysis, verification, and requirements transformation. The automotive industry should thus fully adopt model-based RE.11

Studies have identified model-based development as a possible solution to many ASE challenges. However, automakers haven’t fully exploited its potential because of the lack of both a unified development process and an integrated tool chain.1 To deal with this issue, practitioners have taken a pragmatic, demand-driven, and ad hoc approach to adapting their engineering methods and processes to the available tool environments. As a result, the practice suffers from redundancy, inconsistency, and the lack of automation. To achieve a seamless model-based development environment, practitioners must adopt a comprehensive development methodology based on a well-established modeling theory, a common product model, an appropriate process model, and an integrated tool chain.1

Automakers must also develop and carefully manage traceability across requirements artifacts and the rest of the development process. Previous studies have emphasized the importance of aligning RE and software testing and have proposed mechanisms for their improved traceability, adjustment, and coordination.

Agile Practices
The rapid development and continuous delivery of software have become competitive advantages in the automotive industry. However, automakers have only just begun deploying agile development. Studies show that the key challenges to agile adoption in the industry involve transforming the organizational structure and culture, and achieving a shorter release cycle without compromising quality.12 Other challenges include software reuse with agile practices, applying appropriate quality-assurance measures, and collaborating with suppliers and specialists in other disciplines, such as mechanics. To deal with these issues, researchers propose incremental, stepwise agile adoption.12

The demand for improved and innovative functions is increasing. At the same time, the technologies behind online diagnosis and over-the-air updates have matured. These developments create new opportunities for continuous delivery, effective recall management, and rapid innovation based on real-time data analytics.2 This vision requires the development of infrastructure to support continuous integration (CI) and the continuous delivery of automotive systems.

The main concern in automotive system integration is synthesizing and deploying functions without compromising quality-related attributes such as performance and safety. Studies show that syntactical and technological integration-related challenges are typically addressed during the design phase by using a standardized architecture, while other demanding integration issues are addressed late in the integration process.13 This leads to last-minute chaos and the late identification of integration issues. To avoid these problems, integration should be automated and performed frequently. To integrate as often and early as possible, practitioners should invest in virtualized integration platforms. This makes CI possible during the development process and minimizes dependency on other parts of the system. Because manual integration is an expensive and error-prone process, practitioners should automate and integrate the synthesis and deployment of software functions with CI systems.

CI systems enable more frequent and continuous integration of changes, which increases the need for optimized regression testing (RT). The “retest all” approach isn’t feasible for automotive RT because of the cost and large number of test cases. Thus, practitioners must effectively minimize, select, and prioritize test cases. The software engineering literature has proposed many techniques to improve RT processes, but applying them in the automotive industry remains challenging because source code often isn’t available.14 Thus, alternative approaches that exploit historical data and execution traces seem more appropriate.14

Software Quality
The articles we reviewed emphasized software reliability’s importance and its implications for effective resource allocation, product quality improvement, and release-readiness evaluation. One set of researchers investigated the applicability of commonly used software reliability growth models
(SRGM) in embedded-software projects. Of the eight investigated SRGMs, they found that the Logistic and Gompertz models were the most accurate. They also found that the expected shape of defect inflows helps select the most appropriate SRGM model.

Another study investigated software defect-prediction models, their characteristics, and their applicability throughout the automotive software development lifecycle. This study’s findings can help practitioners choose a defect-prediction model based on its characteristics and the availability of input data.

Modeling systems such as the Simulink graphical programming environment and the Stateflow control-logic tool are now important automotive-development assets. Empirical studies show that clone management of models fosters reuse and improves automotive-system quality and maintenance. Researchers have proposed solutions to help practitioners automatically identify clones in behavioral models. Several case studies have shown these approaches’ effectiveness and scalability.

Variability and Reuse
The automotive industry must fulfill a range of legal requirements while addressing different markets via mass differentiation and customization. Variability can thus become complicated, leading to significant costs and risks. Currently, the industry is dealing with several software-reuse challenges. Issues include the imprecise estimation of economic returns, the need to balance immediate customer needs with long-term benefits, variability-management complexity, quality concerns, organizational matters, the lack of a comprehensive reuse strategy, and intensive software tailoring and optimization.

To establish a successful SPL, practitioners must employ visually informed variability management, standardized architecture, seamless model-driven development, and agile practices. Articles we reviewed support automated approaches for variability identification and management, the automatic diagnosis and debugging of product configurations, SPL test-selection mechanisms, and the automatic generation of modular SPL safety cases.

ASE research appears quite relevant for the automotive industry, as evidenced by the large number of empirical studies. However, there have been fewer comparative studies and practitioner-oriented guidelines. The ASE community’s next quest should be to assess which approaches are most appropriate in specific contexts and explain why this is the case. Answering these fundamental questions could advance existing research while also helping practitioners select appropriate technologies and practices. Obviously, this will require closer collaboration between the automotive and research communities.

References
How Software Is Changing the Automotive Landscape

Hans Aerts and Han Schaminée

From the Editors

In the Impact department article in the July/August 2011 IEEE Software, Hans Aerts and Han Schaminée described how TomTom was a volume leader in consumer navigation systems and how TomTom applied that volume leadership when it entered the automotive market. Here, Aerts and Schaminée present what has occurred over the past six years and predict what will happen in the automotive industry. —Michiel van Genuchten and Les Hatton

VOLUME HAS BEEN a key driver in the automotive and navigation domains. TomTom started as a high-volume supplier in the navigation aftermarket. Because car buyers are no longer prepared to pay high prices for custom in-car navigation solutions and because of widely available smartphone navigation apps, an in-car navigation system is no longer a strong selling point for a car. This means ownership costs for navigation systems must decrease.

TomTom had been able to leverage these costs with its aftermarket solutions, but that market has matured rapidly and is in decline. Thus, the company had to find another way to secure sales volume. Clearly, that volume couldn’t come from the custom-made, fully integrated head units (receivers) that TomTom had been successfully delivering from 2009 onward. The company couldn’t get sufficient volume from projects in which it had to provide an integrated hardware-software solution. There was too little competitive advantage in the hardware supply.

So, TomTom changed from supplying head units to providing navigation software to existing head unit suppliers. Previous competitors became partners who provided hardware and nonnavigation functionality; TomTom provided the navigation application services and maps. This created greater demand for the navigation stack, resulting in a substantial increase of sold licenses (see Figure 1) and a higher market share. The data after mid 2017 in Figure 1 reflects the expected number of licenses from confirmed deals. The figure is indexed such that 2016 is the reference index 100 (that is, the 2016 statistic constitutes the baseline for comparison).

Exploiting Synergies

Just increasing the demand wasn’t enough to provide benefits from the higher volume. If automotive-navigation projects still continued to have a high level of bespoke engineering, there would still be no way to lower costs, except from the fact that the market as a whole was growing. The need existed to exploit many more synergies between
the various automotive projects that employed the navigation stack. This automatically brought requirements such as modularity and configurability to the forefront. But the need for built-in quality was also growing. The bespoke-engineering projects often spent much time on manual testing. The belief was that more modularity and better quality would drastically reduce cycle time and project costs. However, these things are easier said than done.

Nevertheless, the investment in modularity and quality was worthwhile. Project costs and cycle times decreased substantially; see Figure 2. TomTom achieved this without decreasing the diversity of available navigation solutions—they all looked different and ran on a variety of platforms.

In this context, a project is a total work package as agreed upon with an automotive customer. A project starts when the contract is confirmed and ends with the product’s release.

**Modularity**

As with many big software stacks, the navigation stack had grown over the past 10 years into a big, monolithic software beast that was difficult to maintain, customize, and integrate on different platforms. TomTom had designed it primarily for aftermarket products, which had much lower requirements for customization and integration. The stack now had to be made ready to enter the automotive market with a high-volume strategy.

First, TomTom split the UI from the core functionality with the business logic, because the UI often must be customized. The company offers the navigation core as a configurable client library with well-documented interfaces.

Another problem was that there were too many code branches, leading to high maintenance. The new setup immediately introduced a strict one-code-branch-only strategy, which quickly reduced maintenance costs substantially.

**Quality**

The navigation core needed to serve many automotive applications. So, it had to meet the highest automotive quality requirements, regarding not only the number of failures but also regression. This required all the teams to focus on quality.

Many teams found out that defining a threshold value for open defects and then solving defects when that threshold is reached (instead

---

**FIGURE 1.** Growth in the number of TomTom software licenses sold. Changing from supplying both hardware and software to just supplying software resulted in a substantial increase of sold licenses and a higher market share.

**FIGURE 2.** With increased modularity and quality, projects became shorter and cheaper. The number in each rectangle indicates the year and month a project started.
of implementing new features) results in larger throughput. Fixing defects early is cheaper and creates a more stable basis for further development. So, agile methods support defect management in the same way that they promote the benefits of limiting work in progress. Thus, TomTom defined limits for the maximum number of open defects for each team. When the number of defects exceeded the limit, defect fixing had priority over feature development, and the reserved weekly capacity for defect resolution increased. Figure 3 shows the decrease in the weighted number of open defects.

Another measure was heavy investment in test automation as part of every build. Regression could be detected early, resulting in higher quality. This supported continuous integration that allowed for early failure detection and a stable platform for further development.

Finally, TomTom introduced advanced static-code-analysis tools. It measured code quality using TICS (TIOBE Software Quality Framework; www.tiobe.com/tics/tics-framework), which supports the software quality defined in the ISO 25010 standard. TICS covers metrics for code coverage, abstract interpretation, cyclomatic complexity, compiler warnings, coding standards, code duplication, fan-out, and dead code. It analyzes hundreds of projects and keeps track of the top three for small, medium, and large projects. Much of the drive for quality was supported by the teams’ desire to be in the top three of one of the categories. This definitely helped boost the navigation stack’s quality.

Figure 4 shows the results of the static code analysis from the third quarter of 2014 to the fourth quarter of 2016. The dip in 2015 was due to a substantial refactoring of the code base, which created some temporary instability but enabled further improvement of the code quality.

**Productivity**

To cope with the increase in navigation stack customers, TomTom had to shorten the cycle time of new features. As the speed of innovation in the automotive industry increases, innovation cycle time must decrease. Increased innovation also leads to more uncertainty, and again, the best way to address that uncertainty is to implement shorter cycle times. This allows for fast feedback on the developed features.

To achieve shorter cycle times and thus increased productivity, TomTom implemented SAFe (Scaled Agile Framework; www.scaledagileframework.com). Stable, colocated, self-managed teams, rather than constantly changing project teams, delivered a flow of...
features. Initially, the product increment frequency was three months. However, when TomTom’s aftermarket products turned out to require even shorter times to respond to market feedback, TomTom lowered the frequency to two months.

Figure 5 shows the improved productivity in terms of story points per person-week. Story points serve as a unified measure for effort estimation. Because of some organizational-structure changes in 2016, the 2016 statistics aren't comparable and therefore aren't in the figure.

Many people have written about agile development’s advantages. It's often suggested that agility doesn't work in environments with strong customer commitments. We learned that the opposite is true. Agility provides much more transparency regarding the feasibility of meeting the commitment than do more traditional project management methods. Accurate monitoring of the velocity gives you a good feel for the need for corrective measures.

But agile development's promises can become reality only if the leadership style changes. Managing by delegating commitment to the team level disastrously affects transparency, quality, and productivity. Teams have no way to deal with the commitment and uncertainty other than by adding window-dressing, lowering quality, or padding their estimates. In contrast, when teams see that the leadership accepts that uncertainty exists and mistakes will be made, they’ll become more engaged. They’ll learn fast, dare to take risks, and be transparent about them, so that commitments aren’t put at risk.

On the Brink of Big Changes

Some people say the automotive industry will see more innovation in the next 10 years than in the past 100 years; we agree. (An overview of the changes’ impact on software appeared in the July/August 2017 IEEE Software.1) For instance, for many years, the functionality of automotive infotainment systems has been a decade behind that of comparable consumer products. (However, automotive systems’ quality is much better, although consumer systems’ quality is improving rapidly.) It's questionable whether users still accept the wide gaps in functionality. When big software companies enter the automotive industry, they’ll bring unprecedented innovation.2 Connectivity and over-the-air downloadable software will be important enablers for that.

One important trend in the automotive industry is increased automation. Cars will be able to drive more autonomously, primarily to substantially improve safety and reduce fatal accidents. Of course, the autonomous car won’t arrive overnight, but over time, more and more support will be offered to drivers to make their journeys safer and more
enjoyable. In addition, autonomous cars will normally have defined destinations, which will enable innovations in traffic management systems to balance traffic and improve its flow.

Another trend aligned with automation is energy conservation, which will increase as more and more cars move toward electrification.

Connectivity will be another key driver for change. Increased connectivity will not only allow for over-the-air software updates, as we mentioned before, but also provide more opportunities for cars that are always connected. Many navigation functions (such as routing, search, and map display) can execute on a server, and many applications on the server can be connected. As more processing power is available, UIs can further innovate and include, for instance, natural-language processing. And maybe even more important, connected cars will create an enormous amount of data that will enable totally new businesses.

An additional advantage of connectivity is that the more-innovative functions can be implemented on the server side, while the features embedded in the vehicle can be supported by a CPU that remains in the same head unit for 10 years. This hybrid setup could achieve much faster market penetration for innovative features.

Already for some years, TomTom has provided services such as routing, search, and map display, and the application of these services is growing rapidly, as Figure 6 shows. The figure shows indexed growth and costs, where January 2015 is index 100. The peak in costs in early 2016 resulted from a commercial promotion whose success exceeded the limits of a supplier’s contract.

The second peak occurred when TomTom changed the supplier. Currently, an increase in the number of service requests results in a clear decrease of costs per request.

One more important trend is that car ownership will decrease and car sharing will be stimulated. Companies such as Uber have already created a revolution by substantially reducing the cost of mobility, but tons of other opportunities will arise from the connected car.

These trends will ensure that carmakers move away from the traditional vehicle and become mobility providers. The key technology will be software rather than the traditional mechanics and electronics that still dominate the automotive industry. Car manufacturers will likely become user centric rather than car centric. If they don’t do that, big software companies will provide mobility services based on standard vehicle platforms. These companies will also create business models in which people pay for the use of mobility services rather than for ownership.

**The Cloud and the Value Chain**

On one hand, pay per use rather than pay for ownership is attractive for companies because it creates recurrent revenues. On the other hand, it makes the business models more sensitive to volume. In the past, business cases for feature development were based on pretty reliable predictions of car sales. In the future, they’ll be based on the expected user behavior, which will be more cost-sensitive. Already, customers prefer to buy a feature such as map updating when they buy a new car rather than renew a subscription each year for a fraction of the cost.

From another viewpoint, the cost models show more variation. Whereas once the cost of the in-vehicle CPU was paid when the car was sold, in the future, each request...
to an online service will create cost. Whereas software designers in the past had to be conscious of the hardware constraints, these days they’ve been less concerned because hardware resources seem unlimited. But now, all of a sudden, they have to care about the server-side costs.

The increased flexibility, offered primarily by connectivity, will allow for more innovation. But, as we mentioned before, innovation comes with more uncertainty, and companies should organize themselves for it. And that doesn’t mean just implementing an agile process. It also requires, for instance, totally reconsidering how the automotive industry wants to deal with its suppliers. This relationship will move from contract execution to a collaboration in which the partners equally share the business risks. It’s still true that the best way to manage these risks is to leverage volume.

In 2011, we concluded that navigation had become a volume game. That will be even more the case in the connected world.

References
Hands-on laboratory exercises help students internalize knowledge so that it can be applied in new contexts. Hence, cybersecurity educators try to create lab exercises that allow students to explore systems, yet provide sufficient guidance so that students achieve the desired learning objectives without becoming lost in minutiae. Challenges associated with developing such exercises include creating and supporting each lab, ensuring students do their own work, and grading exploratory activities.

In many cases, providing labs is difficult because access to physical lab computers—or remote access to institutionally or other centrally provisioned and managed resources—is not practical. Binding students to centralized servers can make self-paced, intermittent activity more difficult, yet, lacking institutional IT equipment and staff, instructors may not be able to present easily managed and deployed fine-tuned lab environments. However, if students run lab exercises directly on their own computers, other problems arise: the results produced may vary from student to student depending on software installed on the computer used, and all the tools required for an exercise may not even execute on certain platforms. The solution is tailored cybersecurity lab environments that eliminate divergent results caused by software differences. This can be achieved by providing students with virtual machine (VM) images containing lab-related software. Students then run VMs on their personal computers or on institutionally provided computers. Through use of VMs, variations in the results among students can be largely limited to hardware performance differences.

But, use of VMs on student computers has several drawbacks. First, exercises involving two or more networked computers require multiple VMs, the hosting of which is beyond the performance capabilities of many student computers. Also, different labs may rely on mutually incompatible configurations or software packages, thus requiring students to either perform complex provisioning steps or to install separate VMs for each lab. The provisioning and administration of the execution environments required by different labs can become a significant distraction and source of frustration for both students and instructors.

Another challenge is that, regardless of how they are provisioned, cybersecurity lab exercises are often susceptible to students' sharing solutions and cribbing from each other's lab reports. Use of VM images on individual student computers complicates schemes designed to verify student performance of their own lab exercises, for example, logging...
and audit features that might be part of a remotely accessed cyber range. Student actions on VMs can be logged; however, use of a single VM for multiple labs would require some method to distinguish the artifacts of different labs. The alternative of allocating each lab to a distinct VM image can be prohibitive in terms of network bandwidth and disk storage on the student computer.

The last challenge is encouraging students to explore the lab environment while providing instructors with a simple way to determine that students have achieved expected milestones. How can students “show their work”? How can instructors observe what students have done and provide advice if they are stuck, yet not have to stand over the students while they complete the entire exercise?

**Labtainers: A Practical Solution Using Docker Containers**

Labtainers is a framework for developing and deploying Linux-based labs involving multicomponent network topologies all hosted entirely on modestly provisioned student computers. Our initial emphasis is on cybersecurity. Docker containers\(^2\) are used to standardize complete lab execution environments, thereby reducing lab setup and configuration distractions. By using containers, labs can incorporate complex topologies without suffering the overhead of running multiple VMs. The Labtainer framework supports automated assessment of student work and allows lab exercises to be individualized for each student, thus discouraging the appropriation of others’ work.

The use of Docker containers simplifies the Labtainer approach to individualizing student labs and recording student activity for later assessment by instructors. The framework automatically collects artifacts from a student lab environment into an archive file that the student forwards to her instructor. Here we describe strategies for ensuring that the artifacts in the archive file are the result of that student’s efforts. We present these strategies in the context of two example Labtainer exercises. The first provides an introduction to network traffic analysis using `tshark`, and the second employs the `nmap` utility to locate a selected network service.

The Labtainer framework supports three types of users. Lab designers are responsible for creating laboratory exercises so that they meet intended learning objectives. Each lab designer determines if and how the lab is parameterized and whether automated assessment will be supported. Instructors assign labs to students and assess their work. Instructors may or may not work with lab designers to create exercises. Students perform the laboratory exercises. They are oblivious to the underlying framework that configures and individualizes their labs and that gathers artifacts required for assessment.

**Target Lab Context and Automated Assessment**

Students start Labtainer exercises by executing a Python script on a Linux host, typically a VM. The script augments the Linux host environment with one or more Docker containers and a set of virtual terminals. Students use the virtual terminals to interact with the containers, which from the students’ vantage point appear to be independent computers. The execution environment within each container is prescribed by the designer of the lab. In the degenerate case where the lab designer provides only a name for the lab, the environment seen by the student will be a `bash` shell on what appears to the student to be an Ubuntu Linux system. The Labtainer framework allows the lab designer to select from a variety of Linux distributions for each container and to include software packages and configuration settings as appropriate for the lab. Designers define virtual networks and the connections among containers. The student sees the resulting network topology and has virtual terminals connected to only those containers indicated by the designer.

After the student performs the lab exercise, she runs another Python script that terminates the lab on the Linux host. This results in the collection of artifacts from her lab activity. She then provides the resulting archive to the instructor. The instructor can review these artifacts on similarly provisioned Docker containers. The framework includes tools for the lab designer to specify expected attributes of the student artifacts, which are then automatically assessed and summarized for the instructor. Labtainer support for consistent execution environment provisioning and automated assessment of student work is described in detail elsewhere.\(^3,4\)

**Attribution through Lab Individualization**

Several approaches to ensuring the individuality of student work are possible in the Labtainer framework: watermarks, per-student artifacts, and per-student solutions. Within a particular lab exercise, these can be used separately or in combination.

**Per-Student Watermarks**

When a student starts a lab, the Labtainer framework incorporates a student-supplied email address into a seed for generation of pseudorandom values. A watermark file is automatically created for each student lab, and this file becomes one of the artifacts in the student’s archive. The watermark value is validated as part of the assessment process initiated by the instructor.
This simple strategy ensures (albeit weakly) that the archive provided by the student originated with the student who started the lab. As will be seen in subsequent sections, the Labtainer framework allows lab designers to improve on the assurances provided by the watermarks.

The simple watermark check inherent in all Labtainer exercises is readily bypassed by replacing a single file in the archive, perhaps by an automated script shared among students, effective on all Labtainer exercises. Variations on the watermark strategy can be defeated by correspondingly advanced automations, for instance, scripts that replace individual artifacts such as the output of a program invocation. These automations become specific to the individual labs and thus require more effort by the benevolent cheater to build and maintain. A fundamental limitation on the robustness of the watermarking strategies is that the Labtainer framework does not keep secrets from the student environment. Our design explicitly avoided the large step in complexity inherent in maintaining secrets.

Additional assurances of the originality of student work rely on choices made by the lab designer. In the Labtainer framework, these schemes typically have one of two purposes. The first, per-student artifacts, provides further evidence that someone performed the exercise on a Labtainer instance that was initiated using the student’s email address. This strategy causes selected artifacts generated by lab exercise steps to be individualized for each student. The second approach, per-student solutions, seeks to ensure that whoever performed the exercise did not simply reproduce dictated actions. In the Labtainer framework, these per-student solutions, when practical for a given lab, provide more assurance that students performed their own work.

Per-Student Artifacts
Introduction of per-student artifacts makes development of cheating automations more challenging, because individual artifacts themselves are tailored to the individual student. A simple example is an exercise that requires the student to display to standard output the content of a student-specific file on a server once a remote shell on the server is obtained. This would defeat an automation that simply inserts unmodified artifacts into the student’s archive. While one can imagine correspondingly sophisticated automations that are informed by the particulars of the lab exercise, at some point, it becomes easier for enterprising students to publish and re-perform exact keystrokes necessary to create the desired artifacts. For some labs, the keystrokes problem can be addressed by per-student solutions.

A less trivial example of per-student artifacts is found in the Labtainer pcap analysis lab, in which students are introduced to basic network traffic analysis techniques using the tshark utility. Each student’s Labtainer environment for this lab includes a pcap file tailored to that student. The pcap file is individualized by truncating a random quantity of “filler” packets from the start of a baseline pcap file. This results in packet numbers that are unique to the student, while the content of the non-filler traffic remains constant for all students. The student is required to display the single packet of a specific invalid login attempt. Hence, the output of the corresponding tshark command will include a student-specific packet number that can be deterministically reproduced by the assessment function in the instructor’s environment.

Lab designers individualize labs using parameterization configuration files containing commands that cause the framework to replace symbols in selected files with random values derived from the student email address. For the pcap analysis lab, the target of replacement is a parameter passed to a utility that truncates the start of the pcap file. This parameterization utility is invoked the first time a student runs the lab, resulting in truncation of the pcap file seen by the student. (Note: students do not have to complete a lab in one sitting, and if the student wishes, the framework allows her to restart a lab from the beginning, with complete reinitialization and consistent personalization.) The configuration file entry shown in Figure 1 causes the symbol “START_FRAME” in the file “fixlocal.sh” to be replaced by a random value between 1 and 100.

Lab designers define automated assessment in assessment configuration files that identify student artifacts and their expected attributes. The pcap analysis lab assessment configuration files identify the standard input of the tshark command as an artifact of interest. The configuration file includes a directive to extract the “frame.number” filter argument provided to tshark. Labtainer assessment configuration file directives allow the designer to symbolically reference symbols named in parameterization configuration files. The expected value of the frame number is derived by subtracting the random value used during parameterization from the value of the frame number as it existed before the pcap file was truncated. The configuration file entry in Figure 2 subtracts from 190 the

Figure 1. Extract from parameterization configuration file.
“frame_num” value found in a student’s artifact and compares this to the random value that resulted from the directive in Figure 1.

Per-Student Solutions
The example pcap analysis lab is susceptible to one student providing another with the precise keystrokes needed to complete the lab—a problem associated with all labs that rely only on per-student results. Per-student solutions defeat rote repetition of keystrokes. An example is the nmap-discovery lab, which presents the student with a fictional scenario in which he is told that he has an account on an SSH server but is given only the server name (not the network address) and his password. The student uses nmap to locate the server IP address and to discover the SSH port number that the IT department had set to an arbitrary value. This exercise is individualized by assigning a random port number, within a range, to each student. Thus, rote keystroke repetition fails to complete the lab.

The parameterization configuration file for the nmap-discovery lab names symbols in Linux system services configuration files. These system files were modified by the lab designer to contain symbolic names in place of the SSH port numbers. These symbolic names are replaced during parameterization. As a result, system networking services on the configured container will listen to the individualized port number for SSH traffic.

In this particular lab, there is no need for assessment configuration directives to reference symbols in parameterization configuration files, because the student could not have SSH’d to the server unless the port number was discovered. This simplifies automated assessment and is in contrast to the previous example in which the assessment configuration files referenced the pcap truncation parameter.

The nmap-discovery lab automated assessment reveals whether the student was able to use SSH to connect to the target server and the number of times the student invoked the nmap command. The assessment configuration file identifies standard output from the SSH command as an artifact of interest—specifically any output that contains a constant string within a file present on the target SSH server. If this string appears in the artifacts, the student is assumed to have discovered the SSH port number.

Beyond the primary motivation of not rewarding rote replays of lab steps, per-student solutions have an advantage from the perspective of the lab designer. Because the parameterization need not be reproduced in the assessment step, the expected results as represented in the student artifacts can be confirmed without reference to any specific parameterized values generated for that student. Although making a tie between a parameterized value and the expected results in the assessment configuration files is often relatively straightforward, it can become tedious and error prone for some exercises. Consider a forensics-oriented lab that requires students to recover a deleted file from a virtual file system. A simple way to individualize the lab is to add a randomly determined number of filler files into the file system prior to creating the files of interest. The filler files force file offsets and inode numbers of the target files to be a function of the student’s email address. Assessing per-student results for this lab (for instance, comparing student-provided inode values with expected values) is challenging because the effects of filler files on inodes and offsets depend on file system implementation vagaries, and these are not easily predicted. The assessment step for this forensics lab need not be concerned with what the values are for any given student; rather it can rely only on whether the deleted file was in fact recovered. Thus, a per-student solution assumes the student could not have recovered the file content without discovering the offset or inode specific to that student.

Status and Availability
More than 35 labs are available to students and instructors in the Labtainer framework (https://my.nps.edu/web/c3o/labtainers). Each includes a student lab manual, and most include automated assessment and perform per-student individualization. The website also includes a developer package for use by lab designers when creating new labs or transitioning existing labs into the Labtainer framework. The containers themselves are hosted on the public Docker hub (https://hub.docker.com) and are transparently loaded onto a student’s computer when the corresponding lab is first started. The “Labtainer Lab Designer User Guide” describes how lab designers can publish their own Labtainer-based labs on the Docker hub, thus making them available to their students.

Future work we hope to pursue on Labtainers includes GUI-based, integrated lab-authoring tools as well as additional features to support instructors. These include HTML-based reporting of student assessment results and integration into learning management systems. We would also like to convert our development and publishing workflow to a collaborative environment to simplify the integration of
contributions by a community of lab designers.

Acknowledgments
This work was supported by NSF grant DUE-1438893. The views expressed in this material are those of the authors and do not reflect the official policy or position of the National Science Foundation, Department of Defense or the US Government.

References

Michael F. Thompson is with Naval Postgraduate School. Contact him at mthomps@nps.edu.

Cynthia E. Irvine is with Naval Postgraduate School. Contact her at irvine@nps.edu.

got flaws?
Find out more and get involved:
cybersecurity.ieee.org
Computer Science Education in 2018

For the January 2018 issue of *Computer*, six senior computer science educators participated in a Virtual Roundtable where they were asked about how universities are preparing students to deal with contemporary IT challenges, including social networking, false information, and other subjects we see in the news today. For this installment of IT Trends, we asked the same six educators—Michael Lewis (College of William and Mary), Keith Miller (University of Missouri–St. Louis), Shiuhpyng Shieh (National Chiao Tung University), Phillip A. Laplante (Pennsylvania State University), Jon George Rokne (University of Calgary), and Jeff Offutt (George Mason University)—three questions about the current state of computer science education, software engineering, and licensing software engineers using a “Body of Knowledge” approach.

What are today’s core classes in computer science (CS) education? Are they generally uniform in most universities and colleges? How do they compare with those in the early days of CS education (1970s and 1980s)?

LEWIS: Looking at various programs, I was a bit surprised to see the extent to which vestiges of the old core classes are still around: discrete math, data structures and algorithms, programming languages, computer organization or computer architecture, and some sort of software development course. There is also a fairly ubiquitous math requirement of two to three semesters of calculus plus linear algebra.

It’s surprising how static the core of the curriculum seems to be, given how much CS has changed over the years. Of course, the content of the core CS classes has changed in varying degrees, and the electives are far more varied and numerous than what we had in the past. Indeed, many of the topics of electives today did not exist in the 1970s and 1980s.

ROKNE: There has not been a general agreement on what exactly is the core of CS, neither today nor in the past. However, one can probably say that today’s core classes in CS are generally focused on procedural programming, data structures, and algorithm analysis supported by courses in mathematics and statistics, and that in the early days of CS there was a greater emphasis on lower-level programming and courses dealing with computer hardware, and fewer courses on data structures and software engineering.

SHIEH: Today’s core classes include programming (algorithms, programming languages, and data structure), mathematics (linear algebra and discrete mathematics), and system design (computer architectures and operating systems). Although the objectives of the core classes remain the same, the content varies significantly in comparison with that of the 1970s and 1980s. It in-
cludes many new techniques we take for granted today, such as multitasking, just-in-time compilation, networking, and artificial intelligence. Moreover, the core classes need to cover new requirements.

**MILLER:** One way to answer this question is to reference the ACM/IEEE Computer Society Computer Science Curricula 2013 (CS2013; www.acm.org/binaries/content/assets/education/cs2013_web_final.pdf). It’s not the only approach to the computing curriculum available on the world stage (for example, the European Union is working on the Bologna Process, which includes computing); however, CS2013 approaches curriculum not on the basis of “classes” but on the basis of “hours.” An hour is meant to be the amount of material covered in an hour of “lecture,” although the CS2013 document takes pains to not endorse lecture as the preferred method of pedagogy.

See Table 1, which summarizes CS2013 and two previous curriculum guidelines from the same group.

<table>
<thead>
<tr>
<th>Knowledge area</th>
<th>CS2013 Tier 1</th>
<th>CS2013 Tier 2</th>
<th>CS2008 Core</th>
<th>CS2001 Core</th>
</tr>
</thead>
<tbody>
<tr>
<td>AL: Algorithms and complexity</td>
<td>19</td>
<td>9</td>
<td>31</td>
<td>31</td>
</tr>
<tr>
<td>AR: Architecture and organization</td>
<td>0</td>
<td>16</td>
<td>36</td>
<td>36</td>
</tr>
<tr>
<td>CN: Computational science</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DS: Discrete structures</td>
<td>37</td>
<td>4</td>
<td>43</td>
<td>43</td>
</tr>
<tr>
<td>GV: Graphics and visualization</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>HCI: Human–computer interaction</td>
<td>4</td>
<td>4</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>IAS: Information assurance and security</td>
<td>3</td>
<td>6</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>IM: Information management</td>
<td>1</td>
<td>9</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>IS: Intelligent systems</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>NC: Networking and communications</td>
<td>3</td>
<td>7</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>OS: Operating systems</td>
<td>4</td>
<td>11</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>PBD: Platform-based development</td>
<td>0</td>
<td>0</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>PD: Parallel and distributed computing</td>
<td>5</td>
<td>10</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>PL: Programming languages</td>
<td>8</td>
<td>20</td>
<td>21</td>
<td>21</td>
</tr>
</tbody>
</table>
Although they aren’t defined as such, it seems clear that several classes familiar to most CS graduates can be identified by locating large numbers of hours in Table 1: algorithms (AL), computer architecture (AR), discrete structures (DS), programming languages (PL), software development fundamentals (SDF), and software engineering (SE) stand out to me. Personally, I agree that these six areas, covered by some collection of courses, would be fundamental for any undergraduate degree in CS. I wouldn’t think these six would be sufficient, but they certainly would be a base upon which to build.

We can go back to 1968 to compare early CS curricula to the 2013 recommendations. An ACM task force in 1968 recommends eight courses in the core curriculum:

1. Introduction to Computing
2. Computers and Programming
3. Introduction to Discrete Structures
4. Numerical Calculus
5. Data Structures
6. Programming Languages
7. Computer Organization
8. Systems Programming

We see strong similarities with the 2013 document, but several noteworthy differences. Calculus, though discussed in the 2013 document, is not explicitly mentioned in the 2013 core, though discrete structures are. Software engineering appears in 2013, but is not explicit in 1968. Perhaps most striking is how many hours in 2013 are given over to topics not explicitly covered in the 1968 core, including social issues and practice, parallel and distributed computing, and intelligent systems. In the decades between 1968 and 2013, it isn’t surprising that new topics gained importance; perhaps it’s more surprising how similar many of the emphases are.

LAPLANTE: I’ve been teaching in computer science and software engineering programs since the mid-1980s. Over the years, I have seen curricular changes to make room for breadth courses unrelated to CS, and to make the programs more accessible to those who are not strong in mathematics. I believe that these changes come at the expense of a deeper understanding of computation.

ABET (the Accreditation Board for Engineering and Technology) accredits CS programs through guidelines provided by the CSAB (Computer Science Accrediting Board)—a joint effort of the IEEE Computer Society and ACM. The current guidelines say that a CS curriculum must have:
Most CS programs comply with these guidelines with little variation.

Notice that the ABET recommendations omit operating systems, compiler theory, automata theory, database theory, and more—courses that were traditionally taught in most CS programs through at least the late 1990s and that remove the “magic” from how computers and software applications really work. The ACM/IEEE Computer Society 2016 curricular recommendations (www.acm.org/education/curricula-recommendations) provide more depth beyond the ABET criteria, but I’m not sure how widely adopted these are.

OFFUTT: ABET accredits most CS programs in the US, and as a result, the requirements, at least in North America, tend to be fairly uniform. Students from the 1980s (like me) still recognize many of the year one through three courses. Courses like introductory CS, introductory programming, data structures, computer organization and assembly, operating systems, computability, and algorithms are still standard. At the higher level, we see newer topics like security, web and mobile app development, game development, and big data analysis alongside standbys like database, graphics, networks, and AI. I just looked at the requirements at my university (George Mason) and the coursework we had in 1980 would come very close to satisfying current requirements. I find that surprising.

“The brightest computer science graduates are often heavily self-taught due to their passion for this area.” Is this statement true? How often do you experience cases where the students know more than their professors?

MILLER: In my experience, that statement is true, although I don’t think it’s always the brightest students who are heavily self-taught. I think it’s often the most passionate students who are heavily self-taught. And this self-teaching is often in specific, mostly concrete areas. Students might know quite a bit about how to program particular machines or systems, even though their understanding of algorithms in general (for example) might not be particularly sophisticated. I don’t think I can quantify how often this happens to me, but surely in a class of 25 undergraduates it happens several times in a semester where at least one student knows some detail about a particular system or programming language that I don’t know (or don’t remember).

LEWIS: I agree that the brightest CS students generally learn a lot on their own or by working with equally bright students. These students often participate in extra-curricular activities such as personal programming projects and contests, hackathons, and various jobs and internships. This plays a critical role in the development of CS students.

In my experience it’s pretty common for students to know more about particular software tools or frameworks than their professors do. Students frequently return from internships with all manner of knowledge such as web programming frameworks. Also, in their personal software projects they end up doing things that faculty might not typically do (such as hacking Bluetooth drivers on cellphones).

On the other hand, CS faculty tend to know more about the science part of computer science than students, so we’re probably worth keeping around.

LAPLANTE: My experience is that the best young computer scientists have a solid undergraduate education that’s supplemented with lots of hands-on experience obtained through some combination of internships, part-time jobs, and self-study. There are so many open source projects to work on, free tools, and low-cost small platforms—such as Arduino and Raspberry Pi—to play with that there’s no excuse for young computer scientists to not have lots of hands-on experience by the time they graduate. The best students take advantage of these opportunities.

As for students knowing more than their professors—in one area or another, my students know more than me all the time. I can’t be an expert in every programming language, development environment, application domain, or piece of hardware. I constantly learn from my students.
SHIEH: In computer security discipline, both attackers and ethical hackers are often self-taught. More and more self-learning resources are now available on the Internet. The knowledge and implementation skills in CS can be easily digitalized and distributed online, and therefore passion can motivate students to polish their skills through self-learning. As CS domain knowledge expands and grows rapidly, it might be a challenge for professors to keep up and in particular to provide cross-disciplinary hands-on experience. For example, Mark Zuckerberg of Facebook took a year to build a smart home that can follow speech commands, control switches, and even tell a joke. However, in my personal opinion, the academic way of thinking and problem solving can still be the key to the success of new technologies.

ROKNE: It’s certainly true that some of brightest computer science students are heavily self-taught. However, their knowledge base tends to have gaps that need to be filled in through more formal education processes. This was exemplified by one of our very successful undergraduate students who was challenged in his graduate work elsewhere by the advanced theoretical CS courses expected of a graduate student. This student had avoided most theoretical courses offered in his undergraduate studies. It’s not difficult for a bright computer science student to know more than a professor if knowledge is measured by detailed knowledge of specific software or hardware products. For a deep understanding of CS, though, professors are seldom challenged by students.

OFFUTT: I’ve only seen two or three students in my 30-year career who are primarily self-taught to be software engineers. Many are self-taught to be programmers, but most were bad programmers. Certainly not engineers. And many students have very high self-efficacy, believing that they already know everything because they’ve written a few Android apps. I see a few seniors surpass their good teachers, and always find that exhilarating. Unfortunately, I see more students surpass teachers because their teachers do not know much.

The IEEE Computer Society has developed a Software Engineering Body of Knowledge (SWEBOK) and a related examination to become a licensed software engineer. Do you see any impact from this now or in the future?

OFFUTT: Not personally. The last time I looked at the SWEBOK it struck me as out of touch with modern software engineering. It looked like something that might have been appropriate in 1995, not 2017. That’s just my opinion, of course.

ROKNE: It’s difficult to assess the impact of the SWEBOK in a CS department because it’s specifically aimed at software engineering (incidentally, a term coined by F.L. Bauer). CS as we think of it encompasses software engineering but includes other disciplines too. Splitting off new departments of software engineering from CS would mean less interaction and cross-disciplinary research in the overall CS area.

LEWIS: I have not seen any impact of this, even though my state (Virginia) was one of those that first asked for a licensure system. It’s interesting how many of the core topics in the SWEBOK are those core CS courses discussed in the first question (discrete math, data structures and algorithms, software development, and computer architecture). This supports the view that the CS curriculum really has an identifiable core.

SHIEH: The SWEBOK contains necessary knowledge in a summarized form. It will be very helpful to new graduates. In this case, the licensing test might be a good way of ensuring the fundamental knowledge of developers.

LAPLANTE: I led the effort to create the professional engineer (PE) licensing exam for software engineers in the US, so I can answer this question from a unique perspective. For several reasons a different body of knowledge, similar to SWEBOK, had to be created for the licensing exam. We put a tremendous amount of effort into surveying hundreds of professionals, creating the BOK, and writing (and maintaining) the exam. The reasoning and process behind the exam are described in “A Principles and Practices Exam Specification to Support Software Engineering Licensure in the United States of America.”

Unfortunately, since its introduction in 2013 there have been few exam takers (less than 100). There are several reasons for this. First, (with some exceptions) deans, department chairs, and
faculty seem generally uninterested in promoting the exam. Secondly, state departments of engineering have not been uniformly requiring software PEs for public works that contain software. Without this requirement, there is little demand for licensed software engineers. Finally, the path to licensure is difficult because candidates must pass the Fundamentals of Engineering exam. This examination is very broad and expects the candidate to be knowledgeable in areas that most engineers would study, but most software engineers would not.

**MILLER:** The issue of licensing software engineers has a long and contentious history. The issue of whether software engineers should, or will, be licensed has certainly not been settled, despite decades of hard work by several organizations.⁴ For a contrary view in the popular press, see “Programmers: Stop Calling Yourselves Engineers.”⁵

Because of the lack of consensus among professional organizations, I don’t think that licensing will be important in the next few years. It might gain momentum if there are many spectacular disasters. Meanwhile, the SWEBOK might improve our understanding of the principles important to our profession, and might have a more immediate and direct effect on curricula and practice.

**CONCLUSION**

Along with the Virtual Roundtable in the January 2018 issue of *Computer*, we hope this column helps readers get a better understanding as to where computer science education is today. We were pleased to see differing viewpoints to what we believe were challenging questions. We leave it to the readers to make judgments based on their own experiences.

**REFERENCES**


**ABOUT THE AUTHORS**

**Jeffrey Voas** is a computer scientist at NIST and an IEEE Fellow. Contact him at j.voas@ieee.org.

**Rick Kuhn** is a computer scientist at NIST and an IEEE Fellow. Contact him at kuhn@nist.gov.

**Celia Paulsen** is an IT specialist at NIST. Contact her at ceolia.paulsen@nist.gov.

**Kim Schaffer** is an IT specialist at NIST. Contact him at kim.schaffer@nist.gov.
Bridging the Adoption Gap for Smart City Technologies: An Interview with Rob Kitchin

Katja Schechtner, MIT Media Lab

The concept of smart cities has become increasingly popular, but it seems more difficult than ever to bridge the gap between what city administrations want and what developers of smart city technologies offer. At the same time, citizens are trying to understand how these technologies will make their lives better.

Rob Kitchin, a professor and European Research Council Advanced Investigator at the National University of Ireland Maynooth, has more than 15 years of experience in addressing various aspects of smart city concepts to create solutions that work for cities and their citizens alike. He is (co)principal investigator of the Programmable City project, the Building City Dashboards project, the All-Island Research Observatory, and the Digital Repository of Ireland. Much of Kitchin’s hands-on work concerns the development of urban dashboards—such as the Dublin Dashboard (see Figure 1)—that seek to collate all of the urban big data produced by city systems, along with traditional statistical and public administration data, and to visualize that data to make it actionable. Building on this knowledge, Kitchin’s team is currently looking at the social, political, and economic implications of creating smart cities as part of a recent European Research Council grant, and how to build more effective city dashboards through a Science Foundation Ireland grant.


It’s with this background in mind that I discussed with him whether architects and planners and electronic engineers and computer scientists have a different understanding of cities and, if so, how we might bridge the gap.

You have argued that dashboards—and smart city initiatives in general—are often underpinned by a naïve instrumental rationality. What do you mean by this, and why is it an issue?

Smart city technologies mostly seem founded on a rationality that supposes that cities, and their various services and functions, can be steered and managed through a set of technical solutions. In other words, the various complex issues facing citizens and city managers can be disassembled into neatly defined technical problems that can be adequately solved through technology. All that is required to understand, manage, and fix urban issues in a rational, logical, and impartial way is a suitable technical kit, sufficient data, and clever algorithms. In this view, urban dashboards provide a set of data levers for steering the management of the city. The problem with this perspective is fourfold.

First, a technical approach reduces city systems and people to relatively simple components and agents. This mostly ignores the metaphysical aspects of human life; subjectivity; and the role of politics, ideology, soft values, social and institutional structures, capital, and culture in shaping everyday living and urban development and governance. As such, it’s overly reductive and anemic in nature. This is exacerbated by a positioning of technology as neutral, objective, pragmatic, and commonsensical, rather than full of choices, values, and politics.

Second, the technical approach frames urban issues in instrumental and practical ways, rather than within a wider normative framework. So smart city technologies aim to solve questions such as how can we optimize traffic? How can we reduce energy usage? How can we more
effectively police an area? How can we increase the efficiency of service delivery? The issues might be framed with respect to notions of sustainability, safety, security, economic competitiveness, consumer choice, and so on, but often in a shallow, limited sense. For example, developers might state that a technology can make a system more sustainable, without saying what “being sustainable” means beyond instrumental targets. There are many conceptions of sustainability, and adopting the principles of different positions might lead to the development of alternative solutions. Smart city initiatives then rarely start with deeper normative concerns with respect to fairness, equity, justice, citizenship, democracy, governance, political economy, and questions such as, “What kind of cities do we want to create and live in beyond a limited instrumental framing?”

Third, the technical approach assumes that technology can fix all of a city’s issues, rather than acknowledging that some issues might be best solved through political or social interventions, collective action, public policy, investment in infrastructure, or citizen-centered deliberative democracy. There is often a “hammer and nail” mentality in the approach adopted—that is, “if one makes hammers, then all problems look like nails.” In turn, technological solutionism promotes technocratic governance that is narrowly and instrumentally focused and works in constrained and constraining ways.

Fourth, a technical approach can often produce what might be termed “sticking plaster solutions.” For example, technical solutions to traffic congestion are often about trying to optimize flow or re-route vehicles. They don’t address the deep-rooted problem that there are too many vehicles using the road system or provide a solution that shifts people onto public transport or encourages more cycling and walking. Similarly, we’re not going to solve homelessness with an app. It’s an issue of social inequalities and often mental health, drug dependency, and social violence. An app might help manage homeless services more effectively, but it’s not going to address the underlying structural causes.

What the instrumental rationality and associated criticisms of technology-led solutions to city issues mean is that urban planners and city managers are sometimes cautious about adopting them. This doesn’t mean that such planners and managers are anti-technology; rather, they want the optimal solution to an issue, which may or may not involve technology or technology working in concert with other solutions. And they want the technology to be open about its underlying ideas, rationalities, logics, and limitations. For example, with the Dublin Dashboard, we have sought to be open about the aims, principles, praxes, and politics of the initiative and to think critically about how the dashboard influences urban governance.

So beyond an instrumental rationality, do architects and planners and

Figure 1. The Dublin Dashboard, which “provides citizens, public sector workers, and companies with real-time information, time-series indicator data, and interactive maps about all aspects of the city” (www.dublindashboard.ie/pages/ContactUs). Rob Kitchin is the principal investigator.
**electronic engineers and computer scientists have a different understanding of cities?**

My impression is that electronic engineers and computer scientists tend to see the city as a set of knowable and manageable systems—or system of systems—that act in largely rational, mechanical, linear, and hierarchical ways. In addition, city systems are largely treated as generic analytical categories with some typical variances, meaning a solution developed for one city can be transferred and replicated elsewhere. And while cybernetic approaches recognize the complexity and emergent qualities of city systems, they’re still understood as being machinic and largely closed and bounded in nature. This system view of cities is a narrow conception of what a city is and how it works.

In contrast, planners and city administrators understand the city as being complex, multifaceted, contingent, and open and relational, and full of contestation and wicked problems. They typically see cities as places, not systems. From this position, cities have different histories, cultures, social and community relations, economies, governance structures, institutional structures, politics, legacy infrastructures, political and administrative geographies, and interconnections and interdependencies with other places. Cities have a range of different, often competing, actors and stakeholders—government bodies, public sector agencies, companies, nongovernmental bodies, community organizations, and so on—that have different goals, resources, practices, and structures and that are trying to address and manage various issues. This messiness isn’t well captured in computational logic and is difficult to model, predict, and manage through technocratic governance. Understanding cities from this perspective, it seems clear that smart city technology won’t be a silver bullet to urban issues.

**Unlike scientists and engineers, who are usually excited about new or improved methods and tools for planning, monitoring, and managing cities, there seems to be a lack of interest and excitement on the part of urban planners and architects. Why is this?**

Beyond concerns related to technological solutionism and instrumental rationality, there are a number of reasons that cities are cautious with respect to smart city technologies.

The first reason is risk. A city manager will tell you that his or her job is to provide stability, certainty, and reliability in the delivery of city services. A lot of smart city technology is not mature. That is why there is a boom in what has been called “experimental” or “testbed” urbanism or “living labs.” Technologies are still being developed and tested. They are like drugs in the clinical trial phase. Unless there is a compelling reason to be a first mover, perhaps because a problem is so acute that it’s worth taking a risk, or a city is trying to gain a competitive advantage related to economic development, then the city manager would prefer to exploit the second-mover advantage—that is, the advantage of knowing the system will work in solving a particular problem and improve city services. For example, if city management is going to upgrade 50,000 lampposts to smart lighting, they want to know the system is going to work well and do what was promised. They don’t want a newspaper headline that states, “$15 Million of Taxpayers’ Money Wasted.”

Another reason, which is also related to risk, is trust. Planners and city administrators need to trust that new initiatives will work. They have a long history of purchasing technologies that are costly and don’t always deliver on their promises. This includes the wave of first-order urban cybernetic systems in the 1970s that failed horribly and were widely critiqued and abandoned. In fact, the move toward a technocratic approach at that time created a strong backlash in the planning profession, moving it to a much more collaborative, participatory approach. Some smart city technology aims to foster such a citizen-led approach by crowdsourcing data and opinions and fostering debate, but much of the technology is rooted in second-order cybernetics and other technocratic governance approaches.

A third concern relates to the amount of perceived value for money spent and the return on investment. Many technological solutions are not cheap, and it isn’t always clear what the return on investment will be beyond promises that an issue will be ameliorated in some way. Moreover, it’s clear that the same technology will be cheaper and better—in terms of spec, functionality, performance—in a few years, so it’s difficult to know when to make the initial investment. Many cities are currently operating in a condition of austerity, so finances for new investments are constrained. As such, although some technologies could save the city money over the long term, the city still must find the initial investment capital. This is why so much effort is now being expended on new business models for smart city investments.

Another issue is competing demands. City administrations are responsible for managing a range of infrastructures and services. There are many competing demands for a limited budget, and many of these are statutory obligations. Unless a proposed solution will solve a critical problem, rather than merely offer a nice enhancement, it will have trouble competing for attention and resources. What smart city technology developers versus city administrations view as critical issues can be quite divergent.

In addition, city administrations are overloaded. Many stakeholders underestimate the extent to which they are being bombarded by companies, consultants, lobbyists, academics, and so on, all seeking attention for their smart city technology. It can be difficult to work out from the noise which technology is worth pursuing. City administrations must be selective; they can’t pursue all possible initiatives.
The final issue is inertia and resistance. Like all big organizational entities, city administrations have existing practices and legacy systems and are full of internal politics, fiefdoms, and competing interests. Workers can be reluctant to upset the status quo unless what is being proposed is going to substantially improve existing workflows or provide a better solution. In some cases, these “better solutions” will be resisted, especially if they will lead to substantial job cuts.

To promote an understanding between the two groups, architects and urban planners are usually advised to learn a programming language to understand the thinking of “the other side.” What would your advice be for electrical engineers and computer scientists who would like to work with cities?

Rather than start with your question, let’s start with your initial statement. Why should architects and planners learn how to code? Let’s answer that with two questions: Should patients train as medics to understand doctors? Should the users of smartphone apps learn to code to use those apps? Or should the doctors be able to explain their diagnosis in a way that patients can understand and trust, and should the app be intuitive to use and have suitable help support?

There is almost a “blame the victim” mentality in the argument that architects and planners should learn to code, because electronic engineers and computer scientists can’t make their rationalities, imaginaries, logics, and systems intelligible, or convince people that their solutions are better than others.

Architects’ and planners’ work is to create, build, and plan cities. That involves certain kinds of specialist knowledges that take years of training and experience to develop. Similarly, electronic engineers and computer scientists learn specialist knowledge to produce infrastructure and computational systems. Yes, they have pragmatic knowledge of cities based on living in them, but that doesn’t make them experts with respect to architecture, urban design, planning principles, transport systems, social issues, legal and regulatory conditions, and the long history of various kinds of interventions—both policy and practical—previously used to try and solve long-standing issues. So, should electronic engineers and computer scientists train as architects, planners, and other domain specialisms before they start to create technical solutions for city problems? Should they understand in depth the long history of previously attempted solutions and why they are suboptimal? Or should they just work with people who already have this knowledge?

The solution to the gap in knowledge about how particular specialists approach urban issues is communication and mutual learning, not training to gain the core competencies of the other. It’s about working together in teams. It’s about doing full requirements analysis informed from all sides. It’s about respecting each other’s perspectives and approaches—and there are also significant differences in epistemology, ontology, methodology, and ethos across disciplines—and accommodating different viewpoints and knowledge. In our work on urban dashboards, this is how we work; we have a mixed team of social scientists and researchers who are from computer science, data science, and geoscience, as well as having a partnership with six local governments and their domain practitioners and Ordnance Survey Ireland [the national mapping agency] and the Central Statistics Office.

As such, there is a mix of academic and technical expertise and many years of practical experience of trying to address issues on the ground. In other words, we are trying to blend episteme (scientific knowledge), teche (practical instrumental knowledge), phronesis (knowledge derived from practice and deliberation), and metis (knowledge based on experience). We’re also not just interested in building dashboards but also in asking technical/practical questions related to data quality and data access. How will the dashboards create particular views of the world, how will they be used in practice to make decisions and to do political work, and what will be the consequences and ethics of that work? We use a very plural approach that draws on a range of philosophical positions, which we try to frame more normatively.

So, my advice to electrical engineers and computer scientists is to do two things. First, build interdisciplinary/domain practitioner teams that are genuinely interdisciplinary; not teams that are heavily science/engineering dominated with token social scientists that have very limited roles and responsibilities.

Second, understand the critiques levelled at technical approaches to solving urban issues and constraints that are faced by city managers and try to find ways to accommodate and work around them. My experience is that people working for city governments genuinely want to improve the quality of life of their citizens, institutions, and companies located there. If a smart city solution will help them do that, while also mitigating against constraints, then they are interested. But they have to be convinced that the final solution will work with few unanticipated negative consequences, and the value must be worth the investment.

Katja Schechtner is a visiting scholar at the MIT Media Lab and a visiting professor at the Vienna University of Technology.

Contact her at katjas@mit.edu.

Context Aware Ubiquitous Biometrics in Edge of Military Things

Aniello Castiglione, University of Salerno
Kim-Kwang Raymond Choo, University of Texas at San Antonio
Michele Nappi, University of Salerno and University of Naples “Federico II”
Stefano Ricciardi, University of Molise

EDGE COMPUTING CAN POTENTIALLY PLAY A CRUCIAL ROLE IN ENABLING USER AUTHENTICATION AND MONITORING THROUGH CONTEXT-AWARE BIOMETRICS IN MILITARY/BATTLEFIELD APPLICATIONS. For example, in Internet of Military Things (IoMT) or Internet of Battlefield Things (IoBT), an increasing number of ubiquitous sensing and computing devices worn by military personnel and embedded within military equipment (combat suit, instrumented helmets, weapon systems, etc.) are capable of acquiring a variety of static and dynamic biometrics (e.g., face, iris, periocular, fingerprints, heart-rate, gait, gestures, and facial expressions). Such devices may also be capable of collecting operational context data. These data collectively can be used to perform context-adaptive authentication in-the-wild and continuous monitoring of soldier’s psychophysical condition in a dedicated edge computing architecture.

Context Aware Biometrics

In recent years, context-aware biometric systems have been proposed for a wide range of applications characterized by a strong requirement for adapting to variable and unpredictable operating environments and to changing user’s conditions. This scenario, indeed, is common to many human activities that may be affected by changes in the way they are carried out, partly due to human factors and partly to external factors. According to the original definition proposed by Schilit et al., a context aware system is a system that “adapts according to the location of use, the collection of nearby people, hosts, and accessible devices, as well as to changes to such things over time. A system with these capabilities can examine the computing environment and react to changes to the environment.”¹

In this sense, context is “any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves”.²

Context awareness is a concept that has great potential in biometric applications. It may represent, indeed, a crucial aspect for both performance optimization and operational adaptation of all the main tasks of a biometric system (i.e., capture, extraction, matching, and decision stages), particularly in multibiometrics architectures. The awareness of the specific context in which one of the aforementioned tasks is performed allows task execution to be adapted accordingly, for instance by selecting the best possible approach for that context, which
could not necessarily result the most performing in another context. In this regard, the term “context” should be interpreted in its broadest sense, including all the aspects which may possibly affect biometric applications (e.g., operative conditions, environmental conditions, subject’s motion, sensor efficiency, type of usage, etc.).

Biometric information can be usefully exploited in characterizing user context (e.g., by providing user’s physical/behavioral status). On the other side, contextual data (e.g. environmental data, application status, sensors status, operative conditions) may provide valuable information to improve the main aspects of the authentication/recognition process (accuracy, reliability, robustness to variable environmental conditions, robustness to low-quality samples, template security, etc.). The awareness of relevant context information may be of crucial importance to maximize the performance of either single or multiple biometrics, at the same time reducing sensibility to varying conditions and increasing security in the presence of attacks. This may be implemented according to different strategies: by dynamically selecting the optimal feature extraction method for a given capture condition or the most context-suited feature matching algorithm, or even by balancing “speed” versus “accuracy” according to the operational requirements (high “security” versus low “false rejections”). In multibiometric systems, multiple identifiers can be fused together via a weighting strategy based on the context.

A context-based paradigm has been proposed for improving person authentication accuracy by means of a single identifier, such as facial, gait, fingerprint, gestures, as well as by exploiting multiple biometrics. Other known applications include activity recognition and user’s behavior analysis aimed at enhancing the quality of the services provided by a network of devices located in the surrounding environment. Increasingly, typical consumer ubiquitous computing devices, such as Android and iOS devices, facilitate the acquisition of context dependent information due to the embedded sensors like accelerometers, gyroscopes, magnetometer, and environmental lighting sensors. This allows one to increase the security level while accessing to mobile applications and services, without significant impact on performance.

Figure 1 is an example of a context-aware multibiometrics edge-computing architecture, providing battlefield-level biometric monitoring of human resources and context-adaptive unlocking and control of weapons, vehicles, and other equipment.

**Augmenting Internet of Military/Battlefield Things with Context Aware Biometrics**

Context information may also be valuable to achieve performance optimization and operational adaptation of biometric systems implementing ubiquitous user authentication/monitoring on mobile hardware architectures (e.g., in IoMT and IoBT devices that can function as a smart and mobile cyberweapon). In this scenario, context data may also include information about the surrounding environment or terrain, lighting conditions, soldier physical status (e.g., collected via sensors embedded in the combat suit), and ongoing activity (e.g., in motion or at rest, such as a sniper quietly waiting for a target to present itself), and so on.

This regard, it is worth noting that the “Internet of Things” (IoT) concept and technology which are rapidly spreading in many commercial fields, come directly from this network-centric warfare vision originally developed for military purposes by the US Department of Defense over the last decade. This is not unexpected, since the ability to fully exploit the overall amount of information gathered by a wide set of heterogeneous Internet-connected devices deployed on the future battlefield could possibly make the difference in terms of strategic advantage. According to the IoT paradigm, a set of interconnected “things” (e.g., control components, actuators, sensors, information sources, networks, etc.) interact with each other, with humans and with the surrounding environment to the aim of supporting more informed and reliable battlefield command and control operations.

Context aware biometrics may contribute to fully realize the IoT potential by augmenting the available information exchanged among the various kinds of devices with supplementary physical (e.g., heart rate, body temperature or thermal distribution, etc.), and behavioral (body dynamic patterns, speech patterns, etc.) user data, useful for inferring
physiological and emotional conditions of soldiers on the field which could be valuable for critical situation evaluation, and decisional activity. The capability of evaluating implicit static or dynamic user’s characteristics would arguably lead to a better estimation of human factors, compared to approaches based solely on explicit user-system interaction. On the other side, the detection of a context change and change understanding, determine the “awareness factor” necessary to adapt biometric operations to the changing user-status, environmental conditions, application requirements, and security needs. Examples of operative scenarios include dynamic authorization enforcement, context-adaptive access control, and task-dependent on-demand authentication, enabling the implementation of more accurate authorization policies to regulate the access to sites and resources.

Enabling Context Aware Biometrics through Edge Computing Architecture

IoBT involves the full realization of pervasive sensing, pervasive computing, and pervasive communication, leading to an unprecedented scale of information produced by the networked sensors and computing units. Integrating signals from a diverse and dynamic set of sensors, including static ground sensors and soldiers worn sensors, represents one among the several critical challenges facing the implementation of IoT solutions on a battlefield.

While there are a number of potential benefits in context aware multimodal biometrics systems, the memory footprint associated with multiple context-dependent representations of biometric templates can be significant. This is where cloud computing, and more specifically edge computing, plays a facilitating role. For example, context-augmented biometric data can be captured on the go, and sent for processing on edge devices or the cloud “on demand”. Whenever user authentication is requested through single or multimodal biometrics, existing contextual data are uploaded to the edge devices or the cloud. Upon processing, the edge devices or the cloud will return the best matching biometric template (context-wise); thus, improving the robustness of the system.

FIGURE 1. Schematic representation of context-aware multibiometrics edge-computing architecture, providing battlefield-level biometric monitoring of human resources and context-adaptive unlocking and control of weapons, vehicles, and other equipment.
Since in a real application scenario, the number of connected sensors can possibly result very large and the amount of data produced even larger, the system could result overwhelmed by the volume of data in transit. Intelligent data filtering and throttling by edge devices may represent a valid strategy to address this problem, besides upgrading the network infrastructure to increase its maximum bandwidth. It is interesting to note that a similar challenge has already been faced by commercial networking infrastructure with the worldwide diffusion of mobile communication devices, featuring more and more advanced video capture capabilities. In the commercial world, network bandwidth and quality-of-service challenges are being addressed with the use of high-bandwidth carrier grade network infrastructure. ***

References

ANIELLO CASTIGLIONE is an adjunct professor at the University of Salerno (Italy) and at the University of Naples “Federico II” (Italy). His research interests include security, communication networks, information forensics, and security, applied cryptography. Castiglione has a PhD in computer science from the
KIM-KWANG RAYMOND CHOO holds the Cloud Technology Endowed Professorship in the Department of Information Systems and Cyber Security at the University of Texas at San Antonio. His research interests include cyber and information security and digital forensics. He is a senior member of IEEE, a Fellow of the Australian Computer Society, an Honorary Commander, 502nd Air Base Wing, Joint Base San Antonio-Fort Sam Houston, USA, and has a PhD in information security from Queensland University of Technology, Australia. Contact him at raymond.choo@fulbrightmail.org.

MICHELE NAPPI is currently an associate professor of Computer Science at the University of Salerno and team leader of the Biometric and Image Processing Lab (BIPLAB) and author of more than 160 papers in peer reviewed international journals, international conferences, and book chapters. He is coeditor of several international books. His research interests include pattern recognition, image processing, image compression, and indexing, multimedia databases and biometrics, human computer interaction, and VR/AR. An IEEE Senior Member, he is the president of the Italian Chapter of the IEEE Biometrics Council. Contact him at mnappi@unisa.it.

STEFANO RICCIARDI is currently an assistant professor at the Department of Biosciences of the University of Molise. His main research interests include biometry, virtual and augmented/mixed reality, haptics systems, and human-computer interaction. He is a member of IEEE and GIRPR/IAPR, having coauthored about seventy research papers including international journals, book chapters, and conference proceedings. He has been cofounder of a videogame development team focused on 3D simulations. He serves as external expert for the Research Executive Agency of the European Commission. Contact him at stefano.ricciardi@unimol.it.

PURPOSE: The IEEE Computer Society is the world’s largest association of computing professionals and is the leading provider of technical information in the field.

MEMBERSHIP: Members receive the monthly magazine *Computer*, discounts, and opportunities to serve (all activities are led by volunteer members). Membership is open to all IEEE members, affiliate society members, and others interested in the computer field.

COMPUTER SOCIETY WEBSITE: www.computer.org

OMBUDSMAN: Direct unresolved complaints to ombudsman@computer.org.

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

AVAILABLE INFORMATION: To check membership status, report an address change, or obtain more information on any of the following, email Customer Service at help@computer.org or call +1 714 821 8380 (international) or our toll-free number, +1 800 272 6657 (US):
• Membership applications
• Publications catalog
• Draft standards and order forms
• Technical committee list
• Technical committee application
• Chapter start-up procedures
• Student scholarship information
• Volunteer leaders/staff directory
• IEEE senior member grade application (requires 10 years practice and significant performance in five of those 10)

PUBLICATIONS AND ACTIVITIES

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

Periodicals: The society publishes 13 magazines, 19 transactions, and one letters. Refer to membership application or request information as noted above.

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

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

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

Conferences/Education: The society holds about 200 conferences each year and sponsors many educational activities, including computing science accreditation.

Certifications: The society offers two software developer credentials. For more information, visit www.computer.org/certification.

NEXT BOARD MEETING
7-8 June 2018, Phoenix, AZ, USA

EXECUTIVE COMMITTEE

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

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

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

COMPUTER SOCIETY OFFICES
Phone: +1 202 371 0101 • Fax: +1 202 728 9614
Email: help@computer.org
Los Alamitos: 10662 Los Vaqueros Circle, Los Alamitos, CA 90720 Phone: +1 714 821 8380
Email: help@computer.org

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

IEEE BOARD OF DIRECTORS
President & CEO: James Jefferies
President-Elect: Jose M.F. Moura
Past President: Karen Bartleson
Secretary: William P. Walsh
Treasurer: Joseph V. Lillie
Director & President, IEEE-USA: Sandra “Candy” Robinson
Director & President, Standards Association: Forrest D. Wright
Director & VP, Educational Activities: Witold M. Kinsner
Director & VP, Membership and Geographic Activities: Martin Bastiaans
Director & VP, Publication Services and Products: Samir M. El-Ghazzaly
Director & VP, Technical Activities: Susan "Kathy" Land
Director & Delegate Division V: John W. Walz
Director & Delegate Division VIII: Dejan Milojicic
Keep It Simple: Bidding for Servers in Today’s Cloud Platforms

Prateek Sharma, David Irwin, and Prashant Shenoy • University of Massachusetts Amherst

Dynamically priced spot servers are an increasingly popular platform on which to deploy applications. This article shows the effect of spot server bidding on application cost and availability and discusses bidding strategies and new research directions in cloud resource management and fault tolerance.

Today’s infrastructure-as-a service (IaaS) cloud platforms such as Amazon Elastic Compute Cloud (EC2) and Google Cloud Platform rent computing resources on-demand in the form of virtual machine servers. Benefits of using such platforms include a pay-as-you-use pricing model, the ability to quickly scale capacity when necessary, and low costs due to their high degree of statistical multiplexing and massive economies of scale.

IaaS platforms rent servers under a variety of contract terms that differ in their cost and availability guarantees. The simplest type of contract is for an on-demand server, which a customer can request at any time and incurs a fixed cost per unit time of use. In contrast, spot servers provide an entirely different type of contract for the same resources. Spot servers incur a variable cost per unit time of use, where the cost fluctuates continuously based on the spot market’s instantaneous supply and demand. Unlike on-demand servers, spot servers are revocable — that is, the cloud platform can unilaterally preempt them at any time.

In the case of EC2, the cost and availability of spot servers is governed by an auction mechanism. A customer specifies an upper limit (a bid) on the price they’re willing to pay for a spot server, and EC2 reclaims the server whenever the server’s spot price rises above the bid. Because spot servers incur a risk of unexpected resource loss, they offer weaker availability guarantees than on-demand servers and tend to be cheaper — the average price of spot servers is 10 to 30 percent of that of on-demand servers.

Conventional wisdom has held that careful selection of bid-price is important to balance the cost–availability tradeoff — a high bid might increase costs but also increase spot server availability. Here, we show that spot instance bidding need not be complicated. We analyze empirical price data of more than 1,500 spot markets over a six-month period, and show that a wide range of possible bids have approximately the same intended effect on cost and availability. We show that while careful bid selection doesn’t significantly impact the cost–availability tradeoff, careful spot market selection is important to reduce costs and the effects of revocations.

Based on our analysis, we argue for simple bidding strategies and describe best practices when deploying applications on spot servers. We identify challenges and opportunities in reducing the impact of spot revocations (which are akin to machine failures) on application performance. Our goal is to provide practical suggestions to simplify bidding, and to motivate new directions in cloud computing research.

Spot Instance Bidding
Spot instances allow cloud platforms to gain revenue from surplus idle resources. Amazon EC2 uses a market mechanism to sell this capacity
where users place a bid for servers, and EC2 allocates them if the bid is higher than the spot price, which varies continuously based on supply and demand. When the spot price rises above a user’s bid price, EC2 revokes the servers. EC2 determines the spot price by running a sealed-bid multiunit second-price auction. Note that the underlying supply of surplus servers in the spot pool also changes dynamically, because EC2 might take resources from the spot pool to allocate new on-demand instances. Thus, the spot price changes dynamically both as users submit new bids, and as the spot pool’s capacity changes (see Figure 1).

To use a spot server, users place a single, fixed bid, which represents the maximum hourly price that they’re willing to pay. The bids can range from zero to 10 times the on-demand price. Based on the current bids for the server and the available supply, a spot price is determined by a continuous auction. Because this is a second-price auction, users pay the spot price, which might be lower than the bid. If the market price increases to more than the user’s bid, then the spot instance is revoked and terminated after a small (120 second) warning. The prices for each spot server type (also referred to as a spot market) are independently determined. The combination of different server sizes and geographical regions determines a market, and Amazon runs more than 2,500 spot markets globally.

A low bid means that the user is price-sensitive and is only willing to pay a low price for the spot servers. But a server with a low bid might suffer from low availability and a higher likelihood of being revoked if the market price increases to more than the bid price. Frequent revocations might cause application downtimes, missed deadlines, and decreased performance as the application recovers from revocations, which are akin to machine failures. Thus bidding presents the user with a tradeoff between cost and availability/revocation-rate, which might further impact application performance.

Careful selection of bids via bidding strategies has received wide attention in both research and industry. Bidding strategies have been proposed for minimizing costs with different constraints (such as deadlines) for a wide range of applications (such as MapReduce, scientific computing, and so on). Bidding’s complexity might be one reason why, despite its extremely low prices (70 to 90 percent less than on-demand instances), the spot market has low usage. As we discuss, however, the bidding problem in today’s markets (and possibly in future markets) isn’t particularly important for maximizing performance and minimizing costs using spot servers.

**Effect of Bidding**

To understand the effect of bidding for spot instances, we analyze spot prices over a six-month period from March to August 2015 (and longer periods where stated) of 1,500 spot markets. For ease of exposition, we begin our discussion by analyzing the most popular instance types in the most popular region — Linux instances in the region known as us-east-1.

Bidding strategies optimize the cost–availability tradeoff for spot instances: as a user increases their bid, they might pay more per hour, but their availability also increases. However, spot price data across many markets shows that a wide range of optimal bids exist that essentially yield the same availability for the same cost. This is because the spot prices are spiky. In Figure 1, we see that the price spikes can be almost 10 times those of the on-demand price — the same as the upper bound on the bid price. Thus no matter what the bid, the spot instance will be revoked during these large spikes.

To illustrate, Figure 2a shows a cumulative distribution function (CDF) of availability for instance types in five different markets over our six-month period, where the x-axis is a user’s bid normalized to the on-demand price — that is, 2 is 2 times the on-demand price, and so on. As expected, availability monotonically increases with the bid. However, the CDF has an extremely long tail, and there’s little increase in availability after some bid threshold and only bids that fall within the steep range of the incline yield different availabilities. As the graph shows, this range of bids is quite small, providing only a narrow window where changing a bid will have a significant effect on availability. Thus, availability of spot instances isn’t sensitive to bidding for a large range of bid prices.

The insensitivity of bidding in determining the average cost of spot instances can similarly be seen in
Figure 2b. In this case, the cost on the \(y\)-axis is a fraction of the on-demand cost. The cost is monotonically increasing with the bid amount. However, just as with availability, the cost curve has a long tail, such that higher bids result in little or no increase in cost. This occurs because most markets always have a low and stable spot price, with the average spot price <0.2 times the on-demand price. Just as with availability, bidding has little effect on the cost of spot instances, because there’s no penalty for bidding high due to the auction’s second-price nature.

Finally, the frequency of revocations, as indicated by their mean time between revocations (MTBR), is another important metric, since revocations incur overhead for applications that restart or migrate. Figure 2c shows the MTBR for different bids. The figure shows that MTBR range from tens to hundreds of hours. In addition, the MTBR also have a long tail in all but one market, such that bidding high doesn’t significantly increase the MTBR and a wide range of bids exist with effectively the same MTBR. Regardless of the bid price, revocations are unavoidable when using spot instances.

In addition to the five markets discussed previously, we also analyzed these properties in more than 1,500 spot markets, and found that availability, cost, and MTBR are insensitive to bidding for most markets. Figure 3 is a succinct representation of our findings for the 1,500 markets. We show the length of the range of bids for which the availability, cost, and MTBR are all within 10 percent of the optimal bid. The optimal bid is the bid that yields the highest availability and MTBR for the lowest cost. In EC2, the maximum bid can be 10 times the on-demand price, and thus the maximum bid range is 10. We see from Figure 3 that the bid range length is more than 9 for most markets, with few outliers. This indicates that if we were to pick randomly, more than 90 percent of the bids would be within 10 percent of the optimal.

Based on our analysis, we argue that cloud customers need not employ sophisticated bidding and can instead use simple strategies as follows. First, select the spot server type carefully to reduce revocation risk. Then use a bid price equal to the on-demand price. Diversify when possible by choosing multiple spot server types. And finally, if revoked, migrate the application state to a new spot server in a different market. Next, we discuss several design considerations in implementing such a strategy.

Mitigating Spot Instance Revocations
Applications can use the characteristics of spot markets to minimize their costs and the impact of revocations. Careful spot market selection and using the appropriate fault tolerance policies can
drastically reduce the impact of revocations while also lowering costs.

**Market Selection**

Carefully selecting spot markets, instead of being restricted to a particular server type, can greatly increase the effectiveness of spot servers. For distributed applications, a useful strategy is to use multiple spot markets — that is, servers in different availability zones and of different types (small, large, and so on). We observed that price variations across markets are largely uncorrelated (see Figure 4). In general, revocations in different markets don’t occur at the same time. When deployed on a single market, a price spike results in revocation of all the servers. If instead multiple markets are used, then the application can continue to run on remaining unaffected servers.

**Fault Tolerance**

Fault tolerance policies and migration strategies are key in light of the inevitability of revocations and the availability of multiple markets. We can treat server revocation events as fail-stop failures, and choose the suitable application-specific fault tolerance policy. Checkpointing is a commonly used strategy, and by periodically checkpointing state to network storage, the application can resume from the most recent checkpoint. This periodic checkpoint can be performed either at the system-level using nested virtualization, or by using the application’s built-in checkpointing mechanism.

Spot server revocations come with a small 120-second warning, and this warning can expand the fault tolerance choices available and reduce their overhead. For example, it might be possible for certain applications to react on revocation warning and complete a checkpoint, instead of periodically checkpointing. Thus, there exist research opportunities in determining efficient checkpointing and migration strategies to exploit inexpensive but revocable spot servers.

Finally, we must emphasize that it’s the combination of spot market and fault tolerance policies that determines performance and costs. An application deployed on a single market is more susceptible to failure and thus requires stronger fault tolerance, and potentially incurs a higher performance overhead. Selecting the right market might involve considering its average cost, availability, and MTBR. Tools such as Amazon Spot Bid Advisor (see aws.amazon.com/ec2/spot/bid-advisor) can help users in picking markets. A diversified portfolio of markets could reduce revocation risk, but at a higher cost, because this entails picking uncorrelated markets, which might not have the lowest prices.

-1.5 −1.0 −0.5 0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5

**Figure 4.** Correlation between different spot markets in the us-east-1 region. Darker squares indicate higher correlation.

The analysis of historical spot price data leads us to conclude that bidding can be kept simple in today’s spot markets. Instead, users should carefully select markets and fault tolerance policies for their applications.

Our results are predicated on the nature of current spot prices, which are generally low but with occasional spikes. Increased usage of spot servers might change these price characteristics. If the cost and availability CDFs are no longer long-tailed, then bidding’s importance will increase. However, an increased demand for spot servers might be met with an increase in supply, and the price characteristics might remain unchanged. The second-order effects of increasing spot server usage are thus unclear and remain an open question.

**References**


Prateek Sharma is a PhD student in the College of Information and Computer Sciences at the University of Massachusetts Amherst. His research interests focus on cloud computing. Sharma has an MS in computer science from IIT-Bombay. Contact him at prateeks@cs.umass.edu.

David Irwin is an assistant professor in the Department of Electrical and Computer Engineering at the University of Massachusetts Amherst. His research interests include experimental computing systems with a particular emphasis on sustainability. Irwin has a PhD in computer science from Duke University. Contact him at irwin@ecs.umass.edu.

Prashant Shenoy is a professor of computer science at the University of Massachusetts Amherst. His research interests include cloud computing and green computing. Shenoy has a PhD in computer science from the University of Texas at Austin. He’s an IEEE Fellow and ACM distinguished member. Contact him at shenoy@cs.umass.edu.

This article originally appeared in IEEE Internet Computing, vol. 21, no. 3, 2017.

www.computer.org/silverbullet
*Also available at iTunes
IEEE-CS
Charles Babbage Award

CALL FOR AWARD NOMINATIONS
Deadline 1 October 2018

ABOUT THE IEEE-CS CHARLES BABBAGE AWARD
Established in memory of Charles Babbage in recognition of significant contributions in the field of parallel computation. The candidate would have made an outstanding, innovative contribution or contributions to parallel computation. It is hoped, but not required, that the winner will have also contributed to the parallel computation community through teaching, mentoring, or community service.

ABOUT CHARLES BABBAGE
Charles Babbage, an English mathematician, philosopher, inventor and mechanical engineer who is best remembered now for originating the concept of a programmable computer.

CRITERIA
This award covers all aspects of parallel computing including computational aspects, novel applications, parallel algorithms, theory of parallel computation, parallel computing technologies, among others.

AWARD & PRESENTATION
A certificate and a $1,000 honorarium presented to a single recipient. The winner will be invited to present a paper and/or presentation at the annual IEEE-CS International Parallel and Distributed Processing Symposium (IPDPS).

NOMINATION SUBMISSION
Open to all. Nominations are being accepted electronically at www.computer.org/web/awards/charles-babbage. Three endorsements are required. The award shall be presented to a single recipient.

NOMINATION SITE
awards.computer.org

AWARDS HOMEPAGE
www.computer.org/awards

CONTACT US
awards@computer.org
Take the CS Library wherever you go!

IEEE Computer Society magazines and Transactions are available to subscribers in the portable ePub format.

Just download the articles from the IEEE Computer Society Digital Library, and you can read them on any device that supports ePub, including:

- Adobe Digital Editions (PC, MAC)
- iBooks (iPad, iPhone, iPod touch)
- Nook (Nook, PC, MAC, Android, iPad, iPhone, iPod, other devices)
- EPUBReader (Firefox Add-on)
- Stanza (iPad, iPhone, iPod touch)
- ibis Reader (Online)
- Sony Reader Library (Sony Reader devices, PC, Mac)
- Aldiko (Android)
- Bluefire Reader (iPad, iPhone, iPod touch)
- Calibre (PC, MAC, Linux)
  (Can convert EPUB to MOBI format for Kindle)

www.computer.org/epub
Now there’s even more to love about your membership...

Read all your IEEE Computer Society magazines and journals your WAY on myCS.

▶ ON YOUR COMPUTER
▶ ON YOUR eREADER
▶ ON YOUR SMARTPHONE
▶ ON YOUR TABLET

Introducing myCS, the digital magazine portal from IEEE Computer Society.

Finally... go beyond static, hard-to-read PDFs. Our go-to portal makes it easy to access and customize your favorite technical publications like Computer, IEEE Software, IEEE Security & Privacy, and more. Get started today for state-of-the-art industry news and a fully adaptive experience.

LEARN MORE AT: mycs.computer.org
Your career in the technology field offers you endless possibilities. From the broad spectrum of technology specialization to the higher-earning income potential, your future ahead is very bright. But what if something happens to get in your way?

You’ll face countless professional risks. No matter how well you design it or how accurate your advice is, you can be sued. Whether a claim is frivolous or justified—your reputation, assets and career could be at risk. And it could be costly to defend yourself.

IEEE understands your profession like no other! That’s why through the IEEE Member Professional Liability Insurance Program, IEEE members can access enhanced coverage options specifically designed to help minimize their exposures and protect them against the many risks they face. And the Choice Platform provides you with options from industry-leading insurers.*

Protect your career and your future with this important professional liability coverage today!

Let the IEEE Member Professional Liability Insurance Program Help Minimize Your Exposures.

Features risk management and legal services such as contract review, pre-claim prevention and access to a risk management & claims hotline.

Includes a broad definition of computer systems and information technology-related professional services.

Wide range of benefit limits and deductible options available.

1-800-375-0775 | IEEEinsurance.com/IEEEPL