Editor in Chief: **Diomidis Spinellis**
Athens University of Economics and Business, dds@computer.org

Software-Engineering the Internet of Things

Diomidis Spinellis

ENIAC WAS BUILT during the Second World War, from 1943 to 1945. Many consider it the first electronic, general-purpose, large-scale digital computer. Picture it as a room encompassing 36 racks, three printer panels, a card reader, and a card punch. Each rack used hundreds of vacuum tubes to perform a specific function. Many racks acted as accumulators: they received pulses corresponding to the digits of a decimal number and increased accordingly the number stored in them. Others were more specialized; a 1944 floor plan has racks labeled multiplier, partial product, square rooter, denominator, multiplicand, and so on. Three function table racks, initialized through switches, could be carted around on wheels.

Today we build, connect, and configure most Internet of Things (IoT) systems by linking together their sensor, actuator, and computing nodes through cloud infrastructures, mobile apps, and the sharing of security credentials. Similarly, ENIAC was programmed by setting up function tables and switches and connecting its units in the manner required for solving a particular problem, such as generating sine and cosine tables or computing artillery trajectories and shock wave reflections. As you can imagine, such programming was time-consuming and error-prone.

Then, in 1948 a remarkable thing happened. Inspired by the design of EDVAC (Electronic Discrete Variable Automatic Computer), discussed over summer school lectures at the University of Pennsylvania's Moore School, ENIAC's designers realized they could wire it in a revolutionary way. The wiring wouldn't solve a particular numeric program. Instead, the designers would repurpose some of ENIAC's accumulators so that it would read instructions prescribing what actions to perform from its numeric function tables. Think of this as building a command interpreter by assembling together already existing discrete electronic components. Thus, the new wiring transformed ENIAC into a versatile stored-program computer. Rewiring IoT infrastructures into a general-purpose computing fabric can similarly change how modern computation interfaces with our environment.

A Maze of Problems … and Ways Out

No doubt, a paradigm shift from balkanized IoT applications to an integrated infrastructure in which individual IoT nodes are first-class citizens raises formidable challenges. Start with requirements. In a standalone IoT application, it can be easy to satisfy a major functional requirement—say, home security—by controlling the balance of diverse nonfunctional requirements, such as performance, reliability, and usability. However, when multiple IoT nodes and applications get integrated, diverse requirements will interfere with each other (what happens when a burglar triggers a fire alarm?), requiring difficult prioritizations and multicriteria decision making. Given the fluid nature of IoT systems, many decisions we make today during requirements' elicitation might need to be obtained dynamically as the systems operate.

IEEE Software Mission Statement

To be the best source of reliable, useful, peer-reviewed information for leading software practitioners—the developers and managers who want to keep up with rapid technology change.

 0740-7459/17/$33.00 © 2017 IEEE

Designing and constructing integrated IoT systems will also be tough. Standardization in the IoT area is still in its infancy, requiring complex adaptation layers. Requirements associated with control, privacy, and reliability might dictate design decisions that will be at odds with IoT nodes' processing capacity, power budget, bandwidth, and ubiquity. The rewired ENIAC's 60 "order codes," which we today would call its instruction set, required two decades of innovations in programming languages, operating systems, and databases to provide us with the tools and abstractions we now routinely use to build computing applications. Progress of a similar ambition and scale might well be required to truly harness the IoT.

IoT nodes' long-term maintenance will be another nightmare. Many devices will be embedded into buildings, streetlights, bridges, cars, appliances, and other places with lifetimes at least an order longer than the typical PC, smartphone, or server. The IoT nodes will require regular corrective and perfective maintenance, even as their vendors inevitably switch priorities or go out of business over time. We must therefore come up with ways to smooth the handover of IoT devices between vendors.

Also, maintaining a system made up of hundreds (or millions) of diverse embedded IoT nodes is a completely different ballgame than looking after a monolithic cloud application and its systems software. By now, we have some limited experience in such tasks through component-based ecosystems, such as Node.js and microservice architectures. With IoT maintenance, we must apply and extend this knowledge to devices, longer timescales, and a much larger scope.

Then, consider configuration management, which will entail accommodating diverse stakeholders and dynamically changing systems. In DevOps settings, we're already facing tricky problems when a vendor's configuration clashes with locally implemented changes. This problem will only multiply when systems with multiple vendors evolve over time with configurations being set up by unspecialized and untrained users. Unless the state of the art improves dramatically, we might face the choice between unrealized IoT promises and a mess of conflicting, incompatible configurations that will make the 1990s DLL (dynamic linked library) hell appear like heavenly peace. On the other hand, the capabilities of modern decentralized configuration management systems, container technologies, and package managers might offer us the building blocks of a possible solution.

Nailing down the quality of IoT-based systems will also be an arduous, long-term task. Start with security. By definition, IoT systems will be interconnected and offer access to the physical world—a dream come true for cyberwarriors, spooks, and cyberterrorists. Today, we're witnessing the edifice's cracks as rogues take over IoT devices to launch distributed denial-of-service attacks; tomorrow's attacks might be deadly.

Complaints from people unable to control their IoT-enabled fridges hint that usability will be another potent source of ridicule and problems. Furthermore, the physical-world interfaces of IoT nodes mean that reliability failures, of which current software has too many, can have much more serious repercussions than the inconvenience of an odd lost file or application crash.

CONTACT US

AUTHORS

For detailed information on submitting articles, write for our editorial guidelines (software@computer.org) or access www.computer.org/software/author.htm.

LETTERS TO THE EDITOR

Send letters to

Editor, *IEEE Software*
10662 Los Vaqueros Circle
Los Alamitos, CA 90720
software@computer.org

Please provide an email address or daytime phone number with your letter.

ON THE WEB

www.computer.org/software

SUBSCRIBE

www.computer.org/software/subscribe

SUBSCRIPTION CHANGE OF ADDRESS

address.change@ieee.org.
Please specify *IEEE Software.*

MEMBERSHIP CHANGE OF ADDRESS

member.services@ieee.org.

MISSING OR DAMAGED COPIES

help@computer.org.

REPRINTS OF ARTICLES

For price information or to order reprints, email software@computer.org or fax +1 714 821 4010.

REPRINT PERMISSION

To obtain permission to reprint an article, contact the Intellectual Property Rights Office at copyrights@ieee.org.

A CHANGING OF THE GUARD

After four years of dedicated service, Ipek Ozkaya handed the baton of the *IEEE Software* advisory board chair to Rafael Prikladnicki, whom our readers already know as the editor of the Voice of Evidence column. Ipek will continue to serve as associate editor of the magazine's departments, taking over from Richard Selby, who staffed this post over the past two years. Also, after delivering two years of phenomenal growth in *IEEE Software*'s social media presence, Alexander Serebrenik passed on the corresponding management responsibility to his team member Damian Andrew Tamburri.

IEEE Software advertises new volunteer openings through social media and the SEWORLD mailing list (www.sigsoft.org/resources/seworld.html). Keep an eye out if you want to join our team!

Finally, cooperating to achieve the best overall performance of an IoT system, when each device might be selfishly trying to maximize its own, will also be tricky. We've successfully solved a similar problem in the case of Internet bandwidth allocation, through the design of the TCP protocol. Similarly ingenious approaches might well be required for the IoT.

Inevitably, the problems I outlined will feed into an engineering-management challenge: coordinating multiple stakeholders with conflicting interests. One option might involve restricting the setup of IoT systems to walled gardens with strictly defined standards, processes, and compliance testing. This approach might work in a simple controlled environment, such as a small-business building or plant. However, it will rob us of the innovation that open environments can spur and will likely severely limit IoT applications and their impact.

So, unless a winner-takes-all scenario materializes, a closed system will be unsuitable for environments with multiple stakeholders, such as cities, private residences, large office buildings, and mobility applications. Instead, market-based mechanisms should probably be introduced as a way to reduce the friction between technology, social interactions, and the physical world.

Blighted by budget and schedule overruns followed by unreliable operation, ENIAC's birth was anything but auspicious. However, thanks to the perseverance, ingenuity, and openness of the people behind it, it became a defining milestone for modern computing. With hard work and some luck, the IoT can usher in similar changes to how we interact with the physical world.