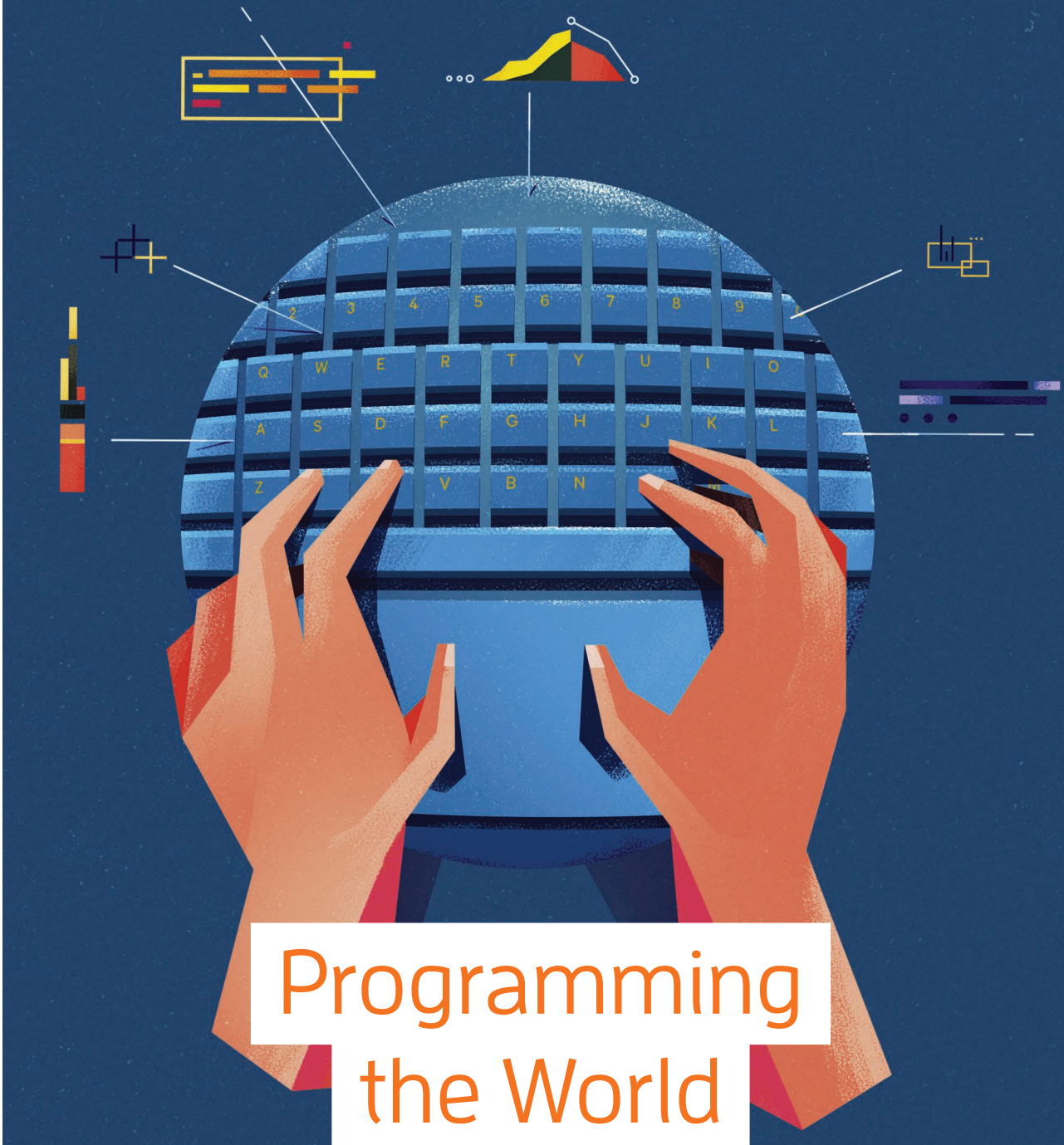

Programming the World

0018-9162/17/$33.00 © 2017 IEEE

Michael Beigl, Karlsruhe Institute of Technology
Florian Michahelles, Siemens Corporate Technology
Hide Tokuda, National Institute of Information and Communication Technology and Keio University
Steve Hodges, Microsoft Research

In the early days of computing, there was a clear separation between the "real" world, in which people manipulated physical objects, and the "virtual" world, in which computers dealt with digital information. Any coupling between these two worlds was largely indirect and manual, making the interface simple to design, build, and operate.

Today, things are different. With the widespread proliferation of embedded systems, the advent of ubiquitous computing, the growth of smartphones (yes, they came later), and now the emergence of augmented reality (AR) and the Internet of Things (IoT), the interfaces are no longer simple in any sense of the word. Computer systems are increasingly integrated with all manner of physical devices in the real world; they must sense, respond to, and control the environment around them. This evolution is set to continue: in the future, potentially anything you can see in the world—and even some things that are too small to see—will contain or be controlled by a computing device.

Going even further, it is possible that computationally controlled materials could contain embedded nanoscale computing elements as the building blocks of future intelligent devices and systems. Researchers are exploring these ideas, with examples including CMU's Claytronics project and MIT Media Lab's Radical Atoms Initiative. Not only do these smart materials come in many physical forms, but they also provide a new type of computational fabric consisting of a mesh of somewhat independently controllable processing elements, providing an even tighter integration between the real and virtual worlds than ever before.

Computing evolution has brought about multiple paradigm shifts in how we use technologies. In the living laboratory that is modern life, our ability to design and control the computers and computing power of the future will require an ability to program systems and applications in situ and in vivo.

As with the first computers, these new computing devices are fundamentally programmable. But to program these new materials is to program the world around us. The processors powering embedded systems, ubiquitous computing applications, AR environments, and the IoT are increasingly taking the form of a computational fabric that we all interact with on a daily basis. Understanding how best to go about developing and controlling these systems—akin to *programming the world* we live in—is the primary inspiration for this special issue of *Computer*.

And who will be the programmers of the future? Today, the vast majority of computer systems are programmed by IT specialists and professional programmers. But this model might not be viable in the future. People from all walks of life will naturally want to customize their digitally controlled environments—to suit their mood, the occasion, and individual needs—just as they have always done in more traditional environments. Sometimes human motivations are clear, practical, and predictable, and other times they are innate, emotional, or even irrational. So the systems that control the world around us must be programmed to respond well to very personal experiences. This naturally suggests that the end users themselves will increasingly become the programmers. However, we cannot expect everyone to develop professional programming skills simply to survive in and enjoy their own "smart" environments.

We have already seen a significant uptake of simplified, end-user–directed programming methods. Today they are often used in somewhat "playful" application areas where simple event-based automation tasks are scripted, although they have also enjoyed much

success as a way of introducing programming to children. Although end user programming schemes can lag behind professional programming tools and environments in terms of sophistication, we expect the paradigm to become increasingly dominant in terms of both acquired ability and widespread adoption. Simplified programming methods for the web and IoT services based on trigger-action programming (TAP) methods are already plentiful. These include IFTTT, Atooma, Bipio, itDuzzit, and Tasker. Educational programming environments like MIT's Scratch, Microsoft's MakeCode, and Google's Blockly are being adopted in classrooms around the world. And many of these systems combine an intuitive programming environment that exposes basic functionality—the "low floor" coined by Seymour Papert—with support for a wide variety and complexity of applications—a "high ceiling." When used in conjunction with one of the many physical computing platforms designed to support tangible interaction, environmental sensing and real-world actuation, such as Arduino, Raspberry Pi, or the micro:bit, they become eminently capable of programming the world.

IN THIS ISSUE

We believe that more of these "standard" building blocks are needed, for both software and hardware. Today the world of Internet-enabled "things" is heterogeneous in every regard—devices, software, services, communication protocols, and standards. Fast, flexible, and accessible programming relies on intuitive, seamless, and robust interoperability between these elements. Given the huge range of application areas we envision—such as smart appliances, smart homes and buildings, smart cars, industrial smart transportation, industrial automation, and responsive environments—there are a great many challenges. Three articles in this special issue present some of the latest ideas and developments in this area.

In "A Semantic Web Approach to Simplifying Trigger-Action Programming in the IoT," Fulvio Corno, Luigi De Russis, and Alberto Monge Roffarello report on EUPont. EUPont provides trigger-action programming, much like IFTTT, but based on an underlying ontological structure. This allows the entities expressed by the programmer to be abstracted, leaving it to the system to implement the rule itself. The result is a highly expressive programming approach that hides technical details from the programmer. The authors investigated how many of the rules in a typical IFTTT scenario are no longer needed when using this method, and showed that nearly 40 percent could become redundant.

In "Semantic Development and Integration of Standards for Adoption and Interoperability," Jack Hodges, Kimberly García, and Steven Ray address a specific problem: with so many formal and ad hoc standards relevant to programmable devices and systems, it can be hard to ensure interoperability. As a result, creating a complete system is more about programming single devices and building lots of conversion modules than "programming the world." The authors aspire to solve this problem with an ontological approach. They show how information formats, communications, and applications like automation and smart buildings can be implemented using technologies such as the Web Ontology Language (OWL).

Finally, in "meSchup: A Platform for Programming Interconnected

ABOUT THE AUTHORS

MICHAEL BEIGL is a professor at Karlsruhe Institute of Technology. His research interests include ubiquitous, pervasive, and wearable computing, and the Internet of Things (IoT). Beigl received a PhD in computer science from the University of Karlsruhe. Contact him at michael.beigl@kit.edu.

FLORIAN MICHAHELLES heads the Siemens Corporate Technology WoT research group. His research interests center on the IoT. Michahelles received a PhD in computer science from ETH Zürich. Contact him at florian.michahelles@siemens.com.

HIDE TOKUDA is president of the National Institute of Information and Communication Technology in Japan, and a visiting professor at Keio University. His research interests include ubiquitous and pervasive computing, the IoT, cyber-physical systems, and smart cities. Tokuda received a PhD in computer science from the University of Waterloo. Contact him at hxt@ht.sfc.keio.ac.jp.

STEVE HODGES leads the Sensors and Devices research group at Microsoft Research Cambridge. His research interests include connected devices, novel sensing and displays, power-aware design, and rapid prototyping. Hodges received a PhD in robotics and computer vision from Cambridge University. Contact him at steve.hodges@microsoft.com.

Smart Things," Thomas Kubitza and Albrecht Schmidt present a playful platform for programming the IoT. Their approach tackles two main problems: first, the complexity of networking a heterogeneity of systems and second, the problem of programming. To address heterogeneity and complexity, the meSchup IoT platform provides a technical framework that readily integrates the prevalent technologies in this space. The system includes a code editor that embodies two new programming concepts: referencing by manipulation (RBM) and sampling by demonstration (SBD). RBM and SBD bridge the gap between the code and the physicality of sensors and their properties. For the end user, TouchCompozr allows simplified trigger-action programming. By manipulating the relevant physical objects, parts of the rules are inferred automatically.

These three papers only scratch the surface of the tremendous research challenges associated with building programmable systems in more accessible and intuitive ways. We are excited to see how this field of research continues to develop. Ultimately, we imagine a future in which everyone is empowered to design, construct, and control a computational system, no matter what their background may be—a future in which we are all programming the world.