Building and Evaluating Ubiquitous System Software

Vinny Cahill
Trinity College Dublin

Armando Fox
Stanford University

Tim Kindberg
Hewlett-Packard Laboratories, Bristol

Brian Noble
University of Michigan

One question we debated when setting out as guest editors was whether to entitle this issue *Everyware*—a name for the new class of software needed to power the world of ubiquitous computing. The title we eventually used—the title of this introduction—reflects the more explicit and pragmatic topic on which we settled. We hope before too long to have everyware literally all around us. But we're not there yet; what we have is more like "some here and a little there," with scant interoperation.

From its inception, plenty of vision has existed in ubiquitous computing. For this issue, we decided to concentrate on how to realize the software infrastructure for that vision—on building and evaluating system software. Ubiquitous computing raises major challenges for system software researchers, mainly because of the heterogeneity and volatility that characterizes it. The set of participating users, hardware, and software in ubiquitous computing environments is highly dynamic and unpredictable. How far have we come in meeting the challenges?

In "*One.world*: Experiences with a Pervasive Computing Architecture," Robert Grimm describes work originating from his doctoral studies at the University of Washington to construct—and learn from the construction of—system support for pervasive applications. The work focuses on system abstractions for volatile ubiquitous systems. Grimm gives us an overview of *one.world* and, more importantly, reports on his experiences in building and writing applications for it. He explains what worked and what didn't and offers insights into the implementation's performance and software complexity.

Published by the IEEE CS and IEEE ComSoc ■ 1536-1268/04/$20.00 © 2004 IEEE

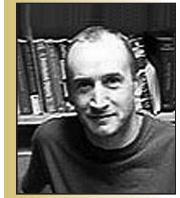
Ubiquitous systems are embedded in our everyday physical world, and it's where the two worlds meet—the physical and the virtual—that some of the most interesting but challenging possibilities arise. This is the subject of the next article, "Semantic Space: An Infrastructure for Smart Spaces," by Xiaohang Wang, Daqing Zhang, Jin Song Dong, ChungYau Chin, and Sanka Ravipriya Hettiarachchi. Designers of context-aware computing systems must deal with the representation of, and computational response to, the physical world. Interestingly, this article draws on work from the Semantic Web community to help solve these problems. Using a "situation-aware" mobile phone, the authors describe how they adapt Semantic Web methods to represent and reason about context and how they can draw on sources of contextual information from Internet services.

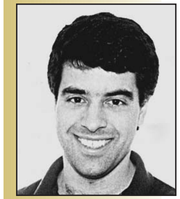
Suppose that researchers or developers want to prototype a ubiquitous computing application, and that their current interest is in the application's functionality—not the specific distributed architecture of its implementation. "Automatic Partitioning for Prototyping Ubiquitous Computing Applications," by Nikitas Liogkas, Blair MacIntyre, Elizabeth D. Mynatt, Yannis Smaragdakis, Eli Tilevich, and Stephen Voida, offers a technique to help. The authors describe how they applied J-Orchestra, a system for partitioning applications into distributable components, to ubiquitous computing. Using a case study of an application that controls peripheral displays, they compare the original, explicitly distributed application with one produced semiautomatically from simpler code with no explicit communication. The authors apply software complexity metrics to demonstrate their approach's benefits and provide an interesting discussion about its limitations.

The final article is "Evaluating a Location-Based Application: A Hybrid Test and Simulation Environment" by Ricardo Morla and Nigel Davies, which describes collaborative work between researchers at Lancaster University and INESC Porto, in Portugal. Even given support for prototyping, constructing a ubiquitous computing application is a daunting task. In general, simulation is a tried and trusted technique for investigating a system's properties without constructing it, but we know little about simulating a ubiquitous system. As we've said, ubiquitous systems are especially challenging and interesting at the cusp between the physical and virtual: a simulator must handle physical events such as location changes as well as distributed system events and network traffic. What's compelling about this work is its open approach. It shows how to use existing Web Services toolkits to combine freely available components, such as a network simulator, with novel components, such as a location simulator and ubiquitous application code.

We started on a note of pragmatism, with a question about progress. Much of the work this special issue describes is early, but so is the research community's work as a whole. What we know about software for ubiquitous systems—the knowledge that we don't already possess from the fields of distributed and mobile systems—merits just a chapter or two in a textbook. But chapters there will be, and it's only through yet more pragmatic work on software construction and evaluation that we will fill those pages. P

the AUTHORS


Vinny Cahill is an associate professor of computer science at Trinity College Dublin, where his research addresses middleware and programming language support for mobile and sentient computing. He received his PhD in computer science from the University of Dublin. Contact him at vinny.cahill@cs.tcd.ie; www.dsg.cs.tcd.ie/~vjcahill.


Armando Fox is an assistant professor at Stanford University and a founder of ProxiNet (now a division of PumaTech), which is commercializing thin-client mobile computing technology developed at UC Berkeley. His research interests include the design of robust software infrastructure for Internet and ubiquitous computing services and user interface issues related to mobile and ubiquitous computing. He received his PhD from UC Berkeley as a researcher in the Daedalus wireless and mobile computing project. He is a member of the ACM. Contact him at fox@cs.stanford.edu; www.cs.stanford.edu/~fox.

Tim Kindberg is a senior researcher at Hewlett-Packard Laboratories, Bristol. His research interests include ubiquitous computing systems, distributed systems, and human factors. He received his PhD in computer science from the University of Westminster. Contact him at timothy@hpl.hp.com; http://purl.org/net/TimKindberg.

Brian Noble is the Morris Wellman Faculty Development Assistant Professor of Electrical Engineering and Computer Science at the University of Michigan. His research centers on software supporting mobile and distributed systems. He earned his PhD in computer science from Carnegie Mellon University. He is a recipient of the NSF Career award. Contact him at bnoble@umich.edu.