Programming environments: A grand challenge

Gregory D. Abowd
College of Computing and GVU Center
Georgia Institute of Technology

We are all familiar with Weiser's eloquent vision of ubiquitous computing (ubicomp) described in the September 1991 issue of *Scientific American*. Over the past few years, attempts to approximate that vision have lead me to believe there is a very important, and mostly unsung, challenge that stands in our way. We need to build more programmable environments. Our computing ancestors leveraged metaphors from the physical world to make computing easier for the masses to use. Now the tables are turned and to make ubicomp easier we need to leverage experience from the computing world and consider the physical world as a programmable input/output environment. Getting there involves overcoming some significant challenges in rethinking what we mean by input and output and how we can provide programmatic control to unleash the creative power of interaction designers of tomorrow.

A lot of the early progress in interactive computing appealed to our understanding and comfort with phenomena in the physical world. We are all comfortable with objects in an office and familiar with actions like filing papers in a cabinet, so it was natural to appeal to that physical metaphor, the desktop, to draw us easily into the world of personal computing. Direct manipulation flourished because it was the creation of a digital world of objects reflecting some of the properties and behaviors of objects in the physical world. We have even gone so far as to create all-encompassing virtual worlds that recreate many of the aspects of the physical world.

Many bemoan the stagnant state of our interactive experience as defined by desktop computing. Weiser did not envy the progress of virtual reality, for he thought VR was anathema to his own vision of ubicomp. While I agree with some of these sentiments (the older forms of interaction are not close to the ubicomp vision of interaction), I envy my colleagues trapped in a 2-D GUI or a head-mounted display. They have built themselves tools that ease the pain of implementation, thus encouraging creative exploration. When liberated from the monotonous technical details of implementation, designers maximize the potential for human interaction with technology. User interface toolkits, builders and programming environments made a world of difference to the variety and quality of the desktop computing experience.

If we borrow the analogy of the toolkit or programming environment for ubicomp, what are the challenges that we face? The fundamental challenges are to redefine what constitutes input and output and how they are tied together to provide for interaction within a physical environment. Input is generalized as sensing followed by interpretation and output as actuation at one or more locations. These new definitions should lead to mechanisms for handling input and output in a programmable physical environment.

What qualifies as input and how is input in an environment is different from input on a personal computer? Input on the PC is mostly explicit activity by the user, largely through keyboard and selection device. While that input language is being extended to include other explicit forms, such as speech received by a microphone or gestures seen by a camera, it is important to consider the implicit forms of input. By implicit, we mean the information describing a situation that can be inferred without requiring any further commentary (i.e., explicit mention) by an individual. By walking into a room, a person implicitly announces her presence. A person engaged in a conversation with another individual is implicitly indicating her lack of availability to anyone else. Humans continuously gather and interpret these implicit cues from the environment and people around them. This implicit situational information is referred to as *context* and there is much active research in the area of context-aware computing.

The challenges that arise with context are many, and they must all be overcome before we can interpret activity in a physical environment as valid input to any "program" we might create to react to those actions appropriately. There is no standard representation for the context of everyday life. There is a small set of actions that are considered the language of explicit action for a PC. No such vocabulary exists for implicit action in the real world. Context, as we have described it, is sensed in the environment, and the capabilities of automated sensing and perception pale in comparison to what a human can do. Interpretation is inherently ambiguous. To handle context as input, we must be able to model ambiguity and carry multiple interpretations of the same sensed phenomenon. Context as input is persistent, meaning it needs to be modeled, retained and adapted over time. A single piece of active context should not be viewed as directed toward a specific application, but rather it should now be considered available for any application. One extreme form of sensing is raw capture, an attempt to record a live experience so that it might be replayed in some synthesized form at a later time.

There is no point to gathering information about the present and the past if it is not used to shape and influence the future. Actuation is the generalization of displaying output. It begins with a generalization of the kinds of visual displays we take for granted now in personal computing, except now the display should take on a variety of sizes and resolution and be available in many places. Initially, this will happen with a growth in the number of dedicated visual displays, but that will be replaced by projection technology that can display on any surface within an environment. Once we increase the number of locations for visual display, the challenge will be in determining the appropriate place for any given piece of information to be displayed. And visual display is only one way to alter or actuate the physical environment. A model for output must encompass all of the ways we can influence the perceivable nature of an environment.

It is a significant challenge to produce a single convincing example of a computationally enhanced environment that gives a glimpse of Weiser's vision of ubicomp. It is indeed a grand challenge to produce an environment in which others can more easily and creatively explore that vision as well.

Gregory Abowd

Gregory D. Abowd (pronounced AY-bowd) is an Associate Professor in the College of Computing at Georgia Tech. His research interests lie in the intersection between Software Engineering and Human-Computer Interaction. Specifically, Dr. Abowd is interested in ubiquitous computing (ubicomp) and the research issues involved in building and evaluating ubicomp applications that impact our everyday lives. In the College of Computing, he is directly involved with research with faculty from Software Engineering, the Graphics, Visualization and Usabilty (GVU) Center and the Georgia Tech Broadband Institute.

Dr. Abowd received the degree of B.S. in Mathematics and Physics in 1986 from the University of Notre Dame. He then attended the University of Oxford in the United Kingdom on a Rhodes Scholarship, earning the degrees of M.Sc. (1987) and D.Phil. (1991) in Computation from the Programming Research Group in the Computing Laboratory. From 1989-1992 he was a Research Associate/Postdoc with the Human-Computer Interaction Group in the Department of Computer Science at the University of York in England. From 1992-1994, he was a Postdoctoral Research Associate with the Software Engineering Institute and the Computer Science Department at Carnegie Mellon University.

In the Fall of 1999, the Georgia Tech Alumni Magazine did a profile on Dr. Abowd and some of his research.

For additional information, see: http://www.cc.gatech.edu/fac/Gregory.Abowd/