Works in Progress

Editor: Anthony D. Joseph ■ UC Berkeley ■ adja@cs.berkeley.edu

Prototypes in Pervasive Computing

EDITOR'S INTRODUCTION

The four entries in this Works in Progress department span different aspects of prototype building for pervasive computing. The first discusses ongoing student projects exploring pervasive computing in automotive environments. The second develops a rapid prototyping environment that combines both form and function. The third explores the challenges of recognizing situations and recurring contextual information, and the fourth investigates approaches to expressing high-level workflow paradigms for device interactions. —*Anthony D. Joseph*

PERVASIVE COMPUTING FOR ROAD TRIPS

Marc Böhlen, Jesse Fabian, Dirk Pfeifer, and JT Rinker, *The MediaRobotics Lab, University at Buffalo*

What happens when you mix site-specific design paradigms from the installation arts with sensing technologies common to pervasive computing? That's the problem students at the University at Buffalo's Department of Media Study were confronted with when asked to imagine new uses for distributed sensors in automobiles.

Long gone are the days when driving was only about efficient travel. Numerous electronic gadgets are transforming car culture. Although some electronic devices such as CD players and DVD viewers foster ambient entertainment, the bulk of an automobile's electronic innards operate unnoticed for control and security. This electronic underbelly offers new opportunities to question the automobile's role in our modern lives. Given our lab's desire to expand technologies toward cultural objects, we designed a semester-long workshop on underrepresented aspects of pervasive computing in automobiles.

Figure 1. Soft toys collect image and vibrational travel data and relay it to a base station.

Our focus wasn't user interaction, usability, or safety issues. Rather, we concentrated on second-order travel events that typically occur on road trips. The road trip—a strange attractor to many North Americans from Lewis and Clark to Jack Kerouac and beyond—is more about experiencing the lay of the land than goal-directed travel. As such, it offers an entry point into car culture.

We supplied students with a prototyping kit that included temperature sensors, 2D accelerometers, miniature color cameras, an OBD-II diagnostic interface, and a microprocessor-based control environment. We invited them to use these tools to react to the experience of being on the road.

One student project explored the possibility of using the changing scenery as an input into a massage seat. The RGB vibrator extracted a histogram of the

Published by the IEEE CS and IEEE ComSoc ■ 1536-1268/05/$20.00 © 2005 IEEE

miniature camera's three main color bands and mapped them to motor commands for a massage seat, resulting in situation-specific massages while driving. Blue colors will generate a lower-body massage, red colors an upper-body massage, and green colors a middle-body massage.

The road trip isn't only about going away, but also returning. In a second project, a student integrated the miniature camera and a 2D accelerometer into two stuffed toy animals (see figure 1). As the car drove around, the camera-enabled toy recorded the optical flow of the changing scenery and the accelerometer-enabled toy registered large bumps on the road. As the vehicle approached home, the toys sent their data via wireless link back to the garage where a screen greeted the returning passengers with a free-form interpretation of the data acquired during the trip.

Our projects are still a work in progress. However, we think that expanding pervasive computing to the unstructured aspects of our lives will result in richer information spaces over time. For more information on our work, see www.buffalo.edu/~mrbohlen/automotive_electronics.html or contact Marc Böhlen at marcbohlen@acm.org.

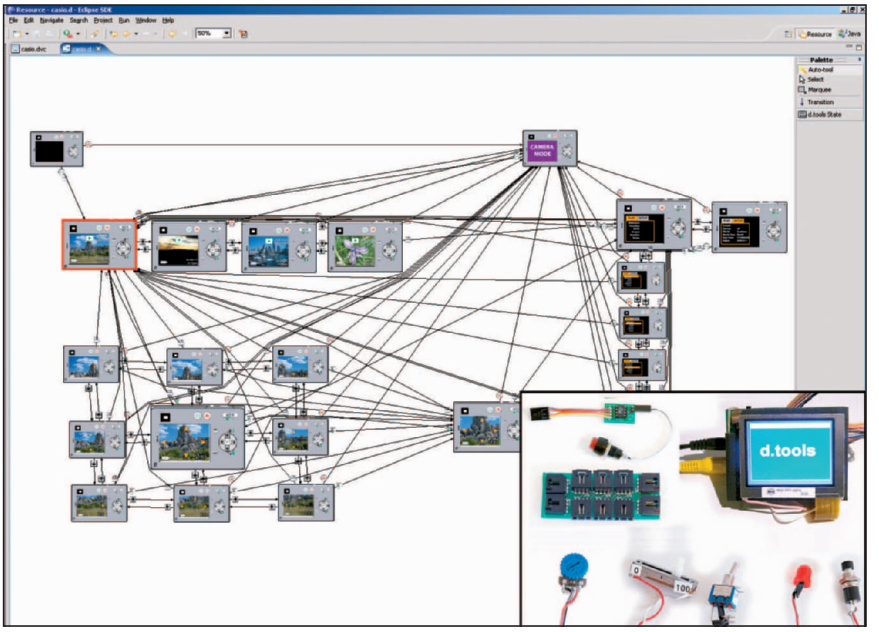

Figure 2. The d.tools visual authoring environment lets designers rapidly create interaction models; the hardware components let them build tangible interfaces for these models.

D.TOOLS: INTEGRATED PROTOTYPING FOR PHYSICAL INTERACTION DESIGN

Björn Hartmann, Scott R. Klemmer, and Michael Bernstein, *Stanford University HCI Group*

In product design, prototypes—approximations of a product along some dimensions—are the essential medium for information, interaction, integration, and collaboration. Information appliances such as mobile phones, digital cameras, and music players are a growing area of ubiquitous computing. Designers currently create two separate sets of prototypes: "looks-like" prototypes that show only the device's form (the atoms) and "works-like" prototypes that use a computer display to demonstrate the interaction (the bits). Because of the required skill and time investment, designers don't build comprehensive prototypes that join form and function until late in development.

At that time, monetary constraints and resource commitments prohibit fundamental design changes. To address this problem, we're creating d.tools. The d.tools integrated design environment lets designers rapidly create prototypes of bits and atoms in concert early in their process. It also fosters shared design conversations and encourages iterations of the design-test-analyze cycle. d.tools is a rough-and-ready tool for creating interactive prototypes as rapidly as designers create paper and foam core prototypes. Our research explores a design tool that's both low-threshold (it's easy to create the prototypes) and high-ceiling (it's possible to prototype complex interfaces). It provides a library of plug-and-play hardware components and a visual authoring environment for creating interaction models (see figure 2). d.tools achieves extensibility through a hardware and software extension architecture based on open standards so engineers and programmers can build upon high-level design specifications.

For more information, see http://hci.stanford.edu/dtools or email Björn Hartmann at bjoern@cs.stanford.edu or Scott Klemmer at srk@cs.stanford.edu.

DESIGN PATTERNS FOR RECOGNIZING SITUATIONS

Seng Loke, *Caulfield School of Information Technology, Monash University*

Pervasive computing applications, from aged care to fire safety, often need to recognize the physical situations of people, devices, and everyday objects. Various sensor hardware and reasoning programs have been employed, from judging human interruptibility[1] to determining an individual's activity or a device's situation.[2] Such work might develop context-aware mobile phones, make appliances safe, disseminate wireless advertisements, or optimize interaction. As such applications proliferate

WORKS IN PROGRESS

and sensing technology advances, the approaches become more diverse. An application might recognize the same or similar situations in different settings or from different perspectives, acquiring the same context information (for example, location) in different ways. Application developers might tackle recurring problems in recognizing situations using a library of solutions (that is, situation patterns) codified in a pattern language. According to the Hillside Group (see http://hillside.net/patterns), patterns "help software developers resolve recurring problems encountered. Patterns help create a shared language for communicating insight and experience about these problems and their solutions."

Our first situation pattern language uses logic programming (which is abstract and enables relationships between sensors), sensor-acquired context, and high-level reasoning.[3] We're also working on creating more general templates for situation patterns akin to design patterns.[4] Such a situation pattern includes the pattern name, the situation to be recognized, the applicability and assumptions concerning the solution, the solution (that is, the sensors and reasoning techniques used), the rationale, known uses and problems, and related situation patterns (for example, a pattern for uninterruptible situations[1]).

For more information, contact Seng Loke at swloke@csse.monash.edu.au.

REFERENCES

1. J. Fogarty et al., "Predicting Human Interruptibility with Sensors," *ACM Trans. Computer-Human Interaction* (TOCHI), vol. 12, no.1, 2005, pp. 119–146.
2. H. Gellersen et al., "Physical Prototyping with Smart-Its," *IEEE Pervasive Computing*, vol. 3, no. 3, 2004, pp.12–18.
3. S.W. Loke, "Representing and Reasoning with Situations for Context-Aware Pervasive Computing: A Logic Programming Perspective," *Knowledge Eng. Review*, vol. 19, no. 3, 2005, pp. 213–233.
4. E. Gamma et al., *Design Patterns*, Addison-Wesley, 1995.

RAPIDLY PROTOTYPING DEVICE ECOLOGY INTERACTIONS

Seng Loke and Sea Ling, *Caulfield School of Information Technology, Monash University*

The American Heritage Dictionary defines ecology as the relationship between organisms and their environment. We envision the computing platform of the 21st century:

- taking the form of device ecologies,
- automating life tasks and work, and
- comprising collections of devices (in the environment and on users) that interact synergistically with one another, with users, and with Internet resources, supported by appropriate software and communication infrastructures ranging from Internet-scale to very-short-range wireless networks.

We're developing a high-level model for describing interactions among devices based on the workflow paradigm,[1] akin to Michael Dertouzos' automation scripts.[2] Users can translate scripts in our high-level language, eco, into a specialized Business Process Execution Language for Web Services specification. They can then execute the scripts via a device ecology workflow, or decoflow, engine. For example, we can translate the script

```
turn on room lights;
close drapes;
show news on television
```

into a decoflow among three devices (the room lights, drapes, and television) and use the decoflow engine to execute it, as figure 3 shows. Each decoflow task might involve a conversation with devices or Internet resources. Web services provide a uniform approach to accessing devices (interfaced as a Web services collection) and Internet resources.

Our system also lets you translate such scripts into Petri net models for analysis to predict their effects on the environment (such as noise level, temperature, and brightness) before execution.[3]

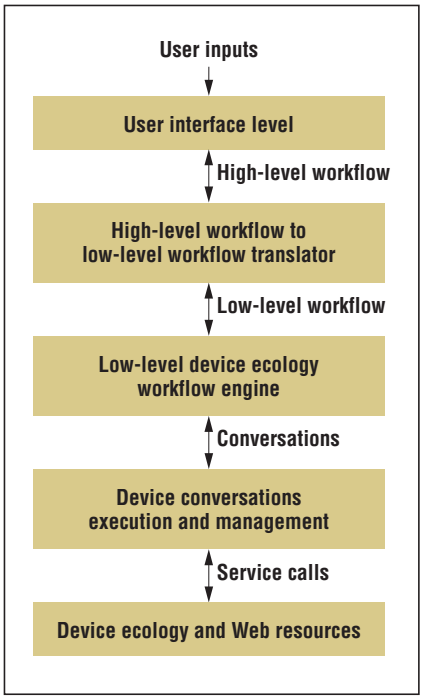

Figure 3. The decoflow engine architecture.

For more information on the model and engine, including further publications, log on to www.csse.monash.edu.au/~swloke/DECO.html or email Seng Loke at swloke@csse.monash.edu.au or Sea Ling at sling@csse.monash.edu.au.

REFERENCES

1. S.W. Loke, "Service-Oriented Device Ecology Workflows," *Proc. 1st Int'l Conf. Service-Oriented Computing*, Springer, 2003, pp. 559–574.
2. M. Dertouzos, *The Unfinished Revolution: How to Make Technology Work for Us—Instead of the Other Way Around*, Collins, 2002.
3. S. Smanchat et al., "Asynchronous and Synchronous Communications in Petri Nets for Run-Time Analysis of a Device Ecology," to be published in *Proc. 7th Int'l Conf. Information Integration and Web-based Applications and Services*, 2005.