d.tools: Visually Prototyping Physical UIs through Statecharts

Björn Hartmann, Scott R. Klemmer, Michael Bernstein, Nirav Mehta
Stanford University HCI Group
Computer Science Department
Stanford, CA 94305-9035, USA
{bjoern, srk}@cs.stanford.edu

ABSTRACT
Processor enabled products, each with its own user interface, have pervaded everyday life. To facilitate interaction design exploration for novel devices, we have developed d.tools, a design tool for prototyping the bits and the atoms of physical user interfaces in concert. It enables designers without specialized engineering or programming knowledge to quickly build functional interactive prototypes. d.tools offers a visual authoring environment that allows for drag-and-drop specification of interaction models for tangible user interfaces in a matter of minutes.

Keywords: Interaction design, prototyping, tangible UIs

INTRODUCTION
We have entered the age of ubiquitous computing: processor enabled products, each with its own user interface, have pervaded everyday life. While interfaces on the desktop have ossified around the WIMP paradigm, interfaces for mobile, ambient, and tangible devices are still evolving through experimentation. Designers of desktop computing applications can rely on a vast library of development aids, but tools that support the design of novel ubiquitous computing devices are just emerging. Current designers of information appliances have to possess expert knowledge in a number of specialized areas, such as programming, embedded microcontrollers, and circuit design in order to build usable prototypes (see FIGURE 1). These requirements make the current development process *solution-information* driven—much time and resources are spent on determining the right implementation—rather than *need-information* driven, where innovation is directed by the desired user functionality.

Von Hippel [6] highlights the importance of appropriate high-level toolkits to separate *need-intensive* design tasks from *solution-intensive* implementation tasks. d.tools empowers designers to perform this need-based innovation. We envision a world where end users can design, program and fabricate their own devices. As a first step in this direction, we are developing a toolkit for product and interaction designers. This user group likely has some knowledge about fabrication, content creation, and interaction design, but does not necessarily possess the engineering expertise to build and program all involved electronic components.

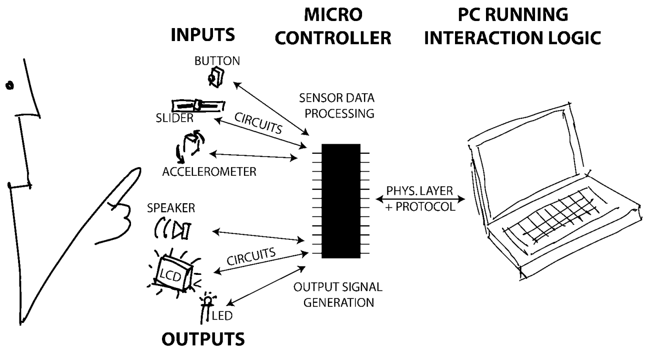

Figure 1: Components of Physical User Interfaces

TOOLS FOR PROTOTYPING
Prototyping is the iterative process of building approximations to reflect on design decisions. It is a central activity of the product design process. Our fieldwork with design professionals and students taught us that in order to be useful, prototyping tools must provide a fast, low-overhead path to generate working results. d.tools achieves this low-threshold requirement by initially insulating users from circuit design and programming APIs through plug-and-play hardware and a visual authoring interface.

In d.tools, designers create interaction prototypes using our PC-based visual authoring environment (see FIGURE 2). The PC can simulate an embedded controller and execute the authored interaction models, communicating with the prototype device through a tether (see FIGURE 3). States in the editor define device outputs; state transitions are triggered by physical inputs. When authoring, designers visually lay out the device's interaction model. This interaction model is executed as a finite state automaton on the PC, processing sensor inputs and generating output signals on the physical prototype. The interaction model is always "live" during the design session, making the hardware ready-to-hand for testing and experimentation. This tight integration of design and execution allows for fast, fluid iterations of the think–design–test cycle [4].

SCENARIO
Consider a product designer assigned the task of developing a handheld media player. She first shapes a rough form prototype out of foam and subsequently embeds d.tools interaction components such as buttons, sliders and an LCD screen. She then connects the device to the PC through our

UIST'05, October 23–27, 2005, Seattle, Washington, USA.

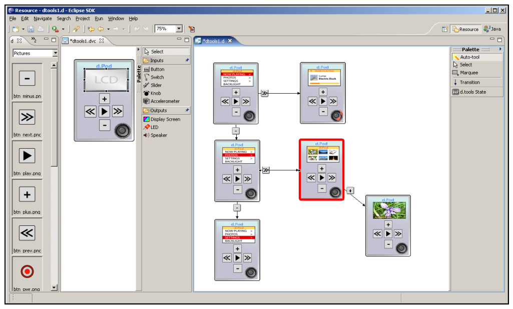
Figure 2: An interaction statechart in our editor

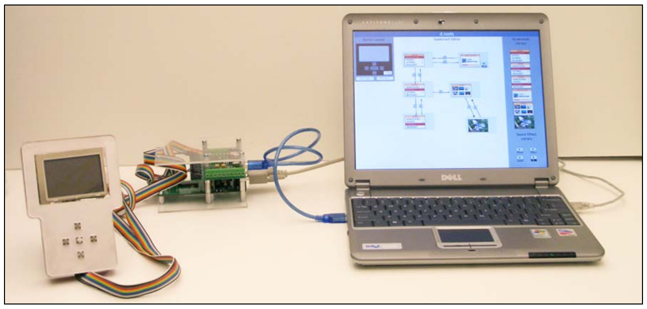
Figure 3: A media player prototype built with d.tools

master controller box. On the PC, she arranges icons depicting the recognized physical I/O components into a virtual representation of the physical device. This iconic representation affords rapid recognition of device elements. The designer then builds an interaction statechart by dragging graphics, sounds or other output specifications onto states and linking states through input transitions. After authoring a few key interaction paths, the designer can hand the functional prototype to her officemate for on-the-spot user feedback.

IMPLEMENTATION
d.tools uses an Atmel ATmega128-based microcontroller board to coordinate communication between the host PC and individual hardware interaction components. The microcontroller communicates with the host PC through OpenSoundControl messages sent over an RS232 connection. The controller board acts as an I^2C master, providing a common multi-drop bus where individual input and output components can be hot-plugged. Each component has an Atmel ATtiny45 controller that runs an I^2C slave program, sending sensor data from an attached input to the master or setting the state of an attached output according to received commands. This polling bus architecture allows the master controller to track presence and identity of hardware components. For graphical output to small screens, d.tools includes an LCD display which can be connected to a PC graphics card with video output (Purdy AND-TFT-25PA-KIT). On the PC, we have implemented the visual state editor in Java as an Eclipse IDE plug-in using the Graphical Editing Framework (GEF). Our design decisions emphasize modularity and extensibility through industry-standard protocols and open source development tool chains.

RELATED WORK
Recent toolkits [1-3, 5] have opened up physical interface development to software developers. However, these tools still require programming experience. Commercial applications such as Max/MSP, a music synthesis and control platform, have shown that domain experts (*e.g.*, musicians) can successfully author complex systems visually without having to be proficient coders. d.tools combines the advantages of these two application classes. Since we use a standard communication protocol between hardware and software, it is possible to connect other hardware platforms such as Phidgets [2] to our visual editor by way of a marshaling service.

WORK IN PROGRESS: EXTENSIBILITY
Having lowered the entry threshold for physical interaction prototyping, we turn our attention to raising the complexity ceiling of devices that can be prototyped. The integration of d.tools into the Eclipse IDE makes it feasible to expose textual programming to users that wish to transcend the possibilities of a purely visual editor.

A hallmark of desktop GUI toolkits is their extensibility: expert users can create their own widgets. Inspired by this, we are developing a high-flexibility hardware architecture. We are researching ways for users to add any device adhering to the I^2C protocol into the d.tools architecture. Using a programmable microcontroller also allows us to run the interaction model on the embedded processor itself and remove the PC tether. The prototype can then be used for field testing outside the confines of a usability lab.

REFERENCES

1 Ballagas, R., M. Ringel, M. Stone, and J. Borchers. iStuff: a physical user interface toolkit for ubiquitous computing environments. CHI: ACM Conference on Human Factors in Computing Systems, *CHI Letters* **5**(1). pp. 537–44, 2003.

2 Greenberg, S. and C. Fitchett. Phidgets: easy development of physical interfaces through physical widgets. UIST: ACM Symposium on User Interface Software and Technology, *CHI Letters* **3**(2). pp. 209–18, 2001.

3 Klemmer, S. R., J. Li, J. Lin, and J. A. Landay. Papier-Mâché: Toolkit Support for Tangible Input. CHI: ACM Conference on Human Factors in Computing Systems, *CHI Letters* **6**(1). pp. 399–406, 2004.

4 Klemmer, S. R., A. K. Sinha, J. Chen, J. A. Landay, N. Aboobaker, and A. Wang. SUEDE: A Wizard of Oz Prototyping Tool for Speech User Interfaces. UIST: ACM Symposium on User Interface Software and Technology, *CHI Letters* **2**(2). pp. 1–10, 2000.

5 Lee, J., D. Avrahami, S. Hudson, J. Forlizzi, P. Dietz, and D. Leigh. The Calder Toolkit: Wired and Wireless Components for Rapidly Prototyping Interactive Devices. In Proceedings of *ACM Symposium on Designing Interactive Systems*. Cambridge, MA: ACM Press. pp. 167–75, August, 2004.

6 von Hippel, E., *Democratizing Innovation*. Cambridge, MA: MIT Press. 2005.