iStuff Mobile: prototyping interactions for mobile phones in interactive spaces

Rafael Ballagas, Faraz Memon, René Reiners, and Jan Borchers

RWTH Aachen University, 52056 Aachen, Germany,
{ballagas,borchers}@cs.rwth-aachen.de
{Faraz.Memon,Rene.Reiners}@rwth-aachen.de

Abstract. iStuff Mobile is a rapid prototyping platform that helps explore novel interactions that combine sensor enhanced mobile phones and interactive spaces. The toolkit includes sensor network platforms, mobile phone software, and a proven rapid prototyping framework. Interaction designers can use iStuff Mobile to quickly create and test novel interactions without making internal hardware modifications to the handset. In this paper, we present the toolkit architecture and provide examples to demonstrate its use.

1 Introduction

Including sensors and actuators in mobile phones enables a variety of new interaction techniques. Interaction designers and researchers need to rapidly build functional interactive prototypes to test these concepts with users early in the design process. However, hardware limitations of current mobile phones may hinder the ability to explore these novel interactions in a cost-effective manner. By lowering the time and financial costs of prototyping such enhanced mobile phone interactions, the number of iterations in the design process can be increased, which has been shown to increase the quality of user interface designs. [14]

iStuff Mobile extends the iStuff toolkit [3] to support rapid prototyping of mobile phone interactions. It allows interaction designers to augment mobile phones with externally attached hardware, such as Smart-Its [6] sensor network modules (See Fig. 1). The framework leverages many of the technologies proven useful in earlier versions of iStuff, such as the Event Heap [10] infrastructure to support distributed application control and the Patch Panel [4] intermediary to promote extensibility and support rapid configuration. Using the iStuff Mobile toolkit, designers can prototype novel sensor-enabled interactions for applications running on the phone such as those described in [8, 17]. Additionally, the toolkit can be used to prototype ubiquitous computing application scenarios such as mobile phone interactions with interactive spaces [16] or public displays [5]. The new contributions of iStuff Mobile are:

– an application that runs as a background service on the mobile phone to expose critical interactive functions, allowing remote control of the foreground application on the phone,

Fig. 1. (Left) A user testing a tilt scrolling interaction with the mobile phone as described in [8]. (Center & Right) back and side views of the mobile phone augmented with the Smart-Its particle board and sensor module. The module can be attached to the phone in whatever position the designer finds most appropriate.

- a new Patch Panel graphical user interface that extends Apple's Quartz Composer [2] and provides an intuitive visual interface to establish relationships between user actions and application feedback on the phone.
- a new graphical front-end for Smart-Its that simplifies configuration of the sensor nodes and integrates Smart-Its to the iStuff framework through a proxy strategy.

The iStuff project has matured from its roots as a toolkit of physical components to an extensible and reusable software framework. As a software framework, iStuff allows third party hardware and software to be rapidly integrated in a ubiquitous computing space and interoperate with existing iStuff-enabled components.

2 Related Work

There have been a variety of hardware toolkits in recent years that help prototype interactions with physical devices. Toolkits like Phidgets [7], Teleo [13], the Calder Toolkit [11], and Smart-Its [6] provide a reusable hardware platform with accessible programming APIs to reduce the barriers of physical device prototyping. However, they do not provide explicit support for prototyping mobile phone interactions. As a software framework, iStuff is conceptually a layer above these toolkits and relies on them to provide the hardware foundation for the mobile phone prototyping solution.

Many of these toolkits have been used to prototype interactions with mobile phones. In the TEA project [17], for example, a predecessor to the Smart-Its platform was used to demonstrate mobile phone context interactions. However,

this project fails to abstract the interface to the mobile phone in a reusable and generic way. Each mobile phone application that used sensor enabled interactions was explicitly programmed to communicate with the sensor board. In iStuff Mobile, the mobile phone applications do not deal directly with the sensor data. Instead, the sensor data is interpreted at the level of the Patch Panel, which in turn issues high-level commands to the mobile phone applications. This abstraction is important to facilitate modification of the relationships between user activity and application feedback without recompiling the mobile phone application. The iStuff Mobile architecture provides an end-to-end prototyping solution that can be used with any mobile phone platform or physical toolkit to support interactions on the phone or using the phone to interact with the surrounding environment.

The d.tools [9] toolkit allows designers to rapidly prototype handheld devices including mobile phones. The d.tools system includes pluggable hardware components and a visual statechart editor to specify interactions. iStuff Mobile is a wireless prototyping solution providing greater freedom of motion in interaction scenarios. The use of real mobile phones makes iStuff Mobile less flexible than d.tools for experimentation with form factor, but the scale of the prototypes is more realisitic. In addition, real devices afford more sophisticated UI design using the Nokia Series 60 [15] software development kit (SDK) or Macromedia Flash Lite [12]. The iStuff graphical programming environment is not restricted to state machines, although the framework does allow them [4]. Finally, d.tools focuses on localized interactions, while iStuff Mobile also supports distributed interactions in ubicomp environments.

3 iStuff Mobile Architecture

iStuff Mobile is designed as a compound prototype architecture (as defined by Abowd et. al [1]) where part of the software is executed on a separate computer as shown in Fig. 2. This compound architecture allows interface designers to prototype interactions that may be beyond the capabilities of current mobile phone hardware. In addition, the compound architecture provides communication capabilities necessary for ubiquitous computing application scenarios. One might argue that a direct communications channel between the sensors and the phone would be more ideal (e.g. through a bluetooth connection). However, this would eliminate the prototyping benefits gained from using the Patch Panel, which allows the relationships between user activity and application feedback to be configured and changed at run-time. It should be emphasized that this toolkit is designed for prototyping and iterative design. Once the design is perfected, it can be reimplemented as an integrated commercial platform.

The relevant components in the system can be distributed across a room, and they communicate through the Event Heap infrastructure. The Event Heap works like a tuplespace with publish-subscribe semantics. This indirect communications model allows clients to communicate without an explicit rendezvous. For components that are not designed to communicate with the Event Heap,

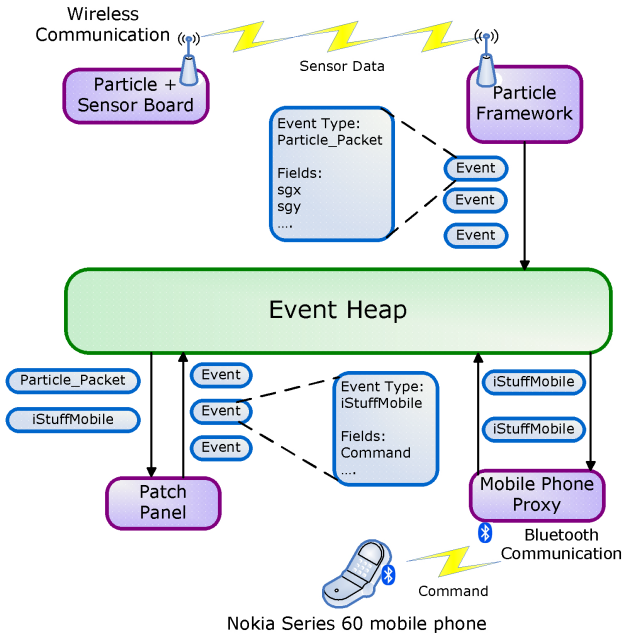

Fig. 2. An illustration with the iStuff Mobile architecture showing the Smart-Its particle framework proxy and the mobile phone proxy. Note that an arbitrary number of diverse hardware and software entities can be used with the Event Heap through proxies and may be connected to each other using the Patch Panel, enabling a broad range of interaction possibilities.

such as Smart-Its and the mobile phone, a proxy strategy can be employed where an external process communicates directly (e.g., through serial ports or application hooks) with the devices to send or receive events on its behalf. This proxy strategy promotes extensibility of the toolkit and simplifies incorporating additional hardware components to the library of reusable building blocks.

In the iStuff architecture, components are not required to share a single event format. Instead, the Patch Panel addresses interoperability by providing an intermediation service. Designers use the Patch Panel to set up relationships between user input and application feedback. These mappings can be set up even if the devices and services were not designed to work together. One of the strengths of the intermediation approach is that these mappings can be specified at run-time without modifying source code of the different interface components involved. To provide a visual front-end to specify these mappings, we have extended Apple's Quartz Composer visual programming environment to include modules that represent the different proxies available for use.

The Smart-Its sensor network platform provides a remote procedure call interface that allows reconfiguration of sensors without modifying the code on the embedded particle boards. In iStuff Mobile, we leverage this capability to develop a cross-platform GUI that combines configuration of sensors, reception of sensor data, and posting of sensor data onto the Event Heap. This GUI allows designers to rapidly configure Smart-Its to work with the iStuff framework.

Designers can remotely execute commands on the phone by sending iStuff-Mobile events to the Mobile Phone proxy, which relays the commands to an application on the phone via a Bluetooth connection. The mobile phone application runs as a background service and relays the commands to the foreground application or the operating system as appropriate (see Fig. 2). The mobile phone application is also capable of sensing user activity, such as key presses, which are relayed to the proxy over the bluetooth connection and subsequently posted as events on the Event Heap. Currently this application is only implemented for the Nokia Series 60 platform, but it can be ported to any modern smartphone platform. This architecture enables interaction designers to prototype interactions with existing applications, or with new ones created with the Series 60 SDK or Macromedia Flash Lite.

4 Prototyping Scenarios

To demonstrate the utility of the iStuff Mobile toolkit, we have used it to prototype a few pervasive computing interactions. For example, we have used the toolkit to prototype improvements to the Sweep interaction [5], which uses camera-based motion estimation to allow the phone to be used as a relative pointing device. The motion estimation algorithm on the low power mobile processor isn't perfect and suffers from some estimation errors. We used the iStuff Mobile toolkit to combine accelerometer data with the camera information to improve the motion estimation as shown in Fig. 3.

Additionally, we have used the toolkit to prototype a multiscreen presentation interaction, as seen in Fig. 4. By pressing a key on the mobile phone, the user is remotely controlling a series of PowerPoint applications that show the current slide as well as the presentation history to the audience. These presentations are controlled through proxies running on different machines in the interactive workspace. Each proxy listens for events with different machine names so that they can be individually controlled.

5 Workshop Goals and Future Work

Our goal for the workshop is to share our experience with this prototyping toolkit and make it available to the PERMID research community. The iStuff Mobile software framework is open source and available at http://istuff.berlios.de. We also hope to learn more about the prototyping techniques of other researchers and create connections between our various research groups for future work. We will work to continue to improve the capabilities of iStuff Mobile, and begin

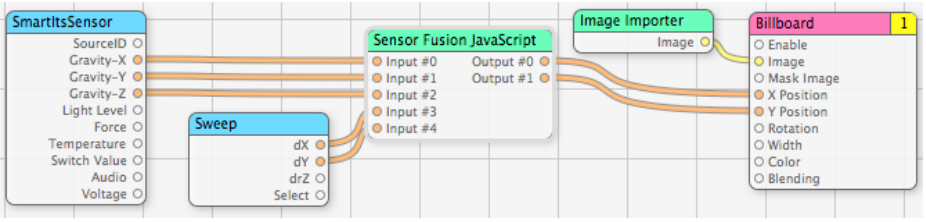

Fig. 3. The Patch Panel GUI implementation for combining accelerometer data with camera-based motion detection to improve the motion detection accuracy. The "Sensor Fusion JavaScript" node implements the algorithm to combine the sensor values in a meaningful way. The JavaScript logic can be modified at run-time to test and refine the sensor fusion strategy. The "Billboard" node displays an image to the screen (e.g. a cursor arrow). The output of the sensor fusion algorithm in the JavaScript node controls the position of the billboard on the screen.

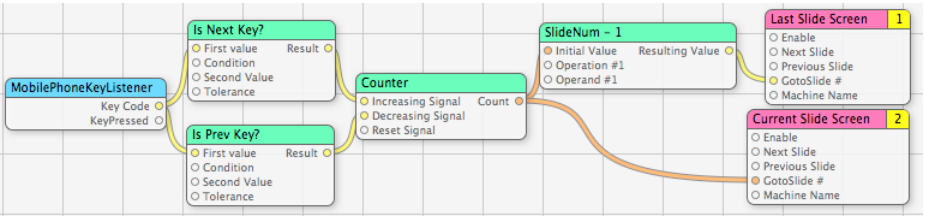

Fig. 4. The Patch Panel GUI implementation for a multiscreen presentation application. On the far left, the "MobilePhoneKeyListener" node receives the key presses from the iStuff Mobile proxy. The two nodes on the far right are iStuff modules to control two instances of the same PowerPoint presentation, each running on a different computer in an interactive workspace.

porting the mobile phone application to different smartphone platforms. We will also continue to refine the Patch Panel user interface. We are considering different approaches to evaluate the toolkit to understand how it impacts the design process.

References

1. Gregory D. Abowd, Gillian R. Hayes, Giovanni Iachello, Julie A. Kientz, Shwetak N. Patel, Molly M. Stevens, and Khai N. Truong. Prototypes and paratypes: Designing mobile and ubiquitous computing applications. *Pervasive Computing, IEEE*, pages 67–73, 2005.
2. Apple Quartz Composer. http://developer.apple.com/documentation/graphicsimaging/conceptual/quartzcomposer/qc_intro/chapter_1_section_1.html.
3. R. Ballagas, M. Ringel, M. Stone, and J. Borchers. iStuff: A Physical User Interface Toolkit for Ubiquitous Computing Environments. In *Proceedings CHI*, pages 537–544, Ft. Lauderdale, FL, USA, April 2003. ACM.

4. R. Ballagas, A. Szybalski, and A. Fox. Patch Panel: Enabling Control-Flow Interoperability in Ubicomp Environments. In *Proceedings PerCom*, Orlando, FL, USA, March 2004. IEEE.
5. Rafael Ballagas, Michael Rohs, Jennifer G. Sheridan, and Jan Borchers. Sweep and Point & Shoot: Phonecam-Based Interactions for Large Public Displays. In *CHI '05: Extended abstracts of the 2005 conference on human factors and computing systems*. ACM Press, April 2005.
6. H.-W. Gellersen, G. Kortuem, M. Beigl, and A. Schmidt. Physical Prototyping with Smart-Its. *IEEE Pervasive Computing Magazine*, 3(3):74–82, July–September 2004.
7. Saul Greenberg and Chester Fitchett. Phidgets: easy development of physical interfaces through physical widgets. In *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 209–218, New York, NY, USA, 2001. ACM Press.
8. Beverly L. Harrison, Kenneth P. Fishkin, Anuj Gujar, Carlos Mochon, and Roy Want. Squeeze me, hold me, tilt me! an exploration of manipulative user interfaces. In *CHI '98: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 17–24, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co.
9. Bjoern Hartmann, Scott R. Klemmer, Michael Bernstein, and Nirav Mehta. d.tools: Visually prototyping physical uis through statecharts. In *Extended Abstracts of UIST 2005*. ACM, 2005.
10. B. Johanson and A. Fox. The Event Heap: A Coordination Infrastructure for Interactive Workspaces. In *Proceedings WMCSA*, page 83. IEEE, 2002.
11. Johnny C. Lee, Daniel Avrahami, Scott E. Hudson, Jodi Forlizzi, Paul H. Dietz, and Darren Leigh. The calder toolkit: wired and wireless components for rapidly prototyping interactive devices. In *DIS '04: Proceedings of the 2004 conference on Designing interactive systems*, pages 167–175, New York, NY, USA, 2004. ACM Press.
12. Macromedia Flash Lite. http://www.macromedia.com/software/flashlite/.
13. Making Things, Teleo. www.makingthings.com/teleo.htm.
14. Jakob Nielsen. Iterative user-interface design. *Computer*, 26(11):32–41, 1993.
15. Nokia Series 60 SDK. http://www.symbian.com/developer/sdks_series60.asp.
16. Trevor Pering, Rafael Ballagas, and Roy Want. Spontaneous marriages of mobile devices and interactive spaces. *Commun. ACM*, 48(9):53–59, 2005.
17. Albrecht Schmidt, Kofi Asante Aidoo, Antti Takaluoma, Urpo Tuomela, Kristof Van Laerhoven, and Walter Van de Velde. Advanced interaction in context. In *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pages 89–101, London, UK, 1999. Springer-Verlag.