Creativity in Software Development in an Academic Research Lab

Andy Wu, Sam Mendenhall, Jayraj Jog, Ali Mazalek
Synaesthetic Media Lab
GVU Center, Georgia Institute of Technology
TSRB, 85 5th St. NW, Atlanta, GA 30318, USA
{ andywu, smendenhall3, jayraj.jog, mazalek }@gatech.edu

ABSTRACT
We review the creative activity in the software development process of the ROSS (Responsive Objects, Surfaces, and Spaces) API. The ROSS API is a tangible toolkit that allows designers and developers to easily build applications for many different tangible platforms while still accommodating the continued evolution of the underlying sensing technologies. We describe the ROSS API specification and discuss its use in application development.

Author Keywords
Software creativity, API, Toolkits, Tangible User Interface

ACM Classification Keywords
H.4.3 Communications Applications

General Terms
Design

INTRODUCTION
Most creativity research on software development focuses on the creativity of solutions during the development period. This paper focuses on three topics in the context of the a novel application programming interface called the Responsive Objects, Surfaces and Spaces (ROSS) API:

- The creative thinking that accompanied drafting the software specification of the API
- The creative methods supported by the ROSS communication architecture that developers can use to achieve software requirements during development
- The creativity involved in developing new applications using the ROSS API

RELATED WORK
There has been a variety of work on developing toolkits for physical sensors, tools to support co-located groupware, and other API frameworks within the areas of ubiquitous computing and tangible interfaces (see [1] for a summary). However there has been little work on toolkits that can integrate this range of interaction environments into a single application programming interface that would allow developers to create truly cross-platform applications. The ROSS API is a software package that extends our previous work in integrated toolkit development [1, 2].

ROSS: RE-THINKING SOFTWARE IN PHYSICAL SPACE
An application programming interface (API) is a software interface designed to enable communication between different software programs. While most software APIs target adjacent levels in the software architecture, ROSS is designed to communicate through several levels across multiple platforms that are very different but have common interactions. ROSS API is designed to establish communication between the hardware driver level to the software application level. To accomplish this, ROSS makes the interface between these levels abstract.

The nested structure of ROSS provides a common abstraction within which application developers can specify the particular configuration of their target platform. This is done using an XML configuration file that describes a given platform's nested structure of spaces, surfaces and objects, as well as its parameters such as ID, type and dimensions. When an application receives such a configuration file from a candidate platform, it can further analyze it and determine whether or not the platform satisfies its needs. This nested structure is best illustrated by an example as shown in Figure 1.

COMMUNICATION ARCHITECTURE
ROSS is designed to integrate different tangible interaction platforms. Developers can take advantage of the ease of the XML format to describe complex nested relationships of objects, spaces and surfaces. ROSS encapsulates all of a user's possible interactions with a device inside of a single representation, which can be fundamentally altered through the XML configuration file. Having the ability to quickly and easily swap out input methods adds an entirely different dimension of creative thinking to application development. Using the toolkit, designers gain a vocabulary for describing different sources of user input, and are free to

Copyright is held by the author/owner(s).
C&C'11, November 3–6, 2011, Atlanta, Georgia, USA.
ACM 978-1-4503-0820-5/11/11.

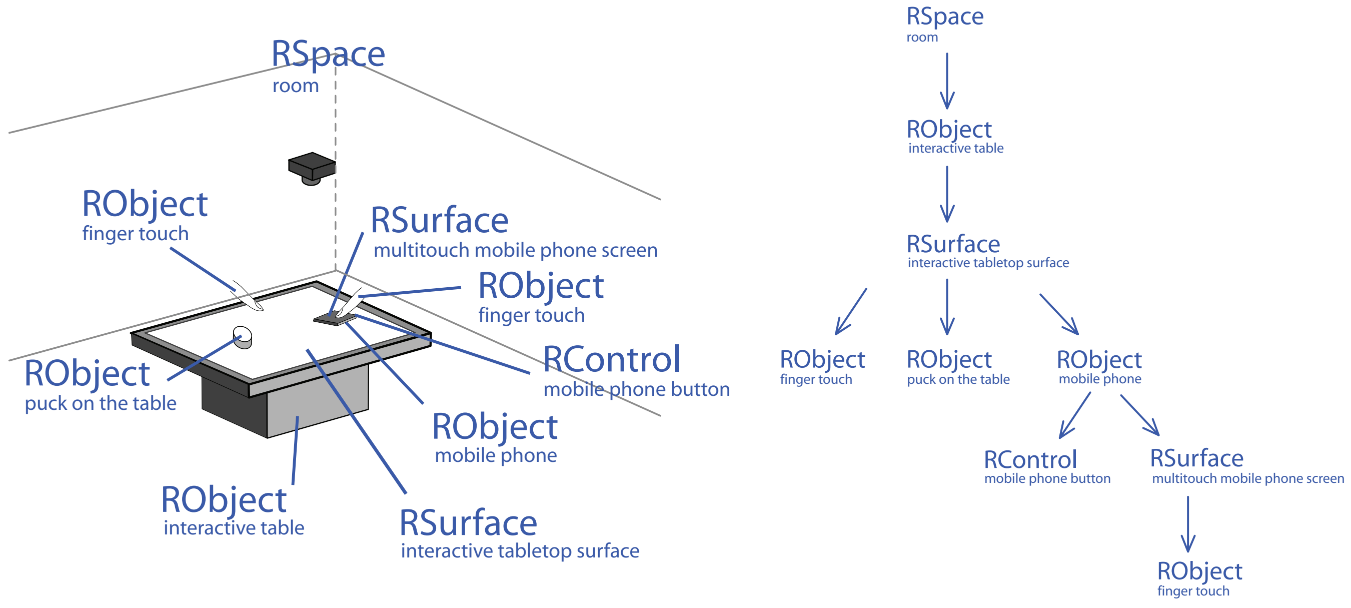

Figure 1. An example of ROSS's nested structure. In the left figure are the ROSS components in a room. There is an interactive tabletop with a puck and a mobile phone on its tabletop surface. Users interact with the tabletop and the objects on top of it. The right figure shows the hierarchical nested structure.

tinker and stretch the boundary between the physical and digital worlds.

APPLICATIONS AND LESSONS LEARNED

ROSS provides abstractions for tangible input devices. In addition, the library allows programmers to treat objects, surfaces and spaces with the same spatial relationships as they have in real life. A number of applications have been created at various stages of the ROSS API development process and demonstrate its flexibility. Examples include: the installation piece *Moons Over You* which tracks multiple people in a space and synchronizes their movements with the display of personalized moons overhead; the ROSS Paint application which allows multiple users to paint a shared drawing across various devices, such as mobile phones, laptops, and wall displays; and ROSS Ripple which demonstrates how the screen space of small devices such as mobile phones can nest into and interact with larger surfaces such as interactive tabletops.

The ROSS API makes programming for these varied platforms simple. It takes care of details such as networking, event handling and device management, allowing programmers to focus on making an engaging application. Some evidence of this is already available. For ROSS Paint and ROSS Ripples, the applications themselves perform very little of the common tasks of establishing connections, managing networking, events and subscriptions, and other necessary tasks. Instead the application code consists entirely of application logic.

Just as sketches and wireframes help GUI designers plan designs and physical models help architects visualize their buildings, we have found that tangibility promotes dialogue between developers and helps them think through design concepts. This is especially true for the ROSS API, since it is designed for integrating tangible user interfaces, many of which are graspable. Usually when we discuss an idea in the lab, we pick up the object and demonstrated it to other colleagues. This characteristic of the "tangible" user interface helps make project conversations more fluid.

SUMMARY AND FUTURE WORK

ROSS uses a fundamentally different design paradigm for specifying a networking API. Through its novel hierarchical nested structure, ROSS closely models the environments for which it is designed. These new concepts create a network of tangible user interfaces that promotes embodied interaction.

The unorthodox API architecture of ROSS opens another window for tangible user interface researchers to implement their applications in a different way. This novel method in API development has not yet been evaluated. Therefore, we cannot make claims about its efficiency. However, once a TUI developer follows the specification defined in ROSS, she can create applications that communicate with heterogeneous devices and benefit from being part of the entire ROSS tangible network.

REFERENCES

1. Mazalek, A. Tangible Toolkits: Integrating Application Development across Diverse Multi-User and Tangible Interaction Platforms. *Let's Get Physical Workshop, DCC'06*, (2006).
2. Reilly, D.F., Rouzati, H., Wu, A., Hwang, J.Y., Brudvik, J., and Edwards, W.K. TwinSpace: an infrastructure for cross-reality team spaces. *Proc. UIST '10*, ACM (2010), 119-128.