The Toolkit / Audience Challenge

David Ledo, Lora Oehlberg, Saul Greenberg
Department of Computer Science
University of Calgary
Calgary, Alberta, Canada, T2N 1N4
{david.ledo, lora.oehlberg, saul.greenberg}@ucalgary.ca

ABSTRACT
A variety of HCI toolkits help designers and developers author particular styles of interactive systems. However, the design, use and evaluation of toolkits are fraught with many challenges. This paper focuses on a subset of challenges that arise from the fit between the toolkit and its intended audience. These challenges include the skill set of that audience, the resources they have, and how they learn. We illustrate these challenges via three toolkits: Phidgets, d.Tools, and the Proximity Toolkit.

Author Keywords
Toolkits; Prototyping Tools.

ACM Classification Keywords
H.5.2. Information interfaces and presentation (e.g., HCI): Miscellaneous; Prototyping.

INTRODUCTION

Toolkits are means of encapsulating design concepts to help a developer realize particular styles of interaction design without undue effort [2]. The developer chooses a toolkit to design within HCI genres and/or to exploit interaction techniques. Broad genres include GUIs, physical / tangible interfaces, and ubicomp. Interaction techniques are narrower, such as gesture recognition and input sensing. Toolkits range in how they can be accessed [3], and can include:

- ***traditional programming***, usually through coding via a functional or object-oriented API;
- ***coding support tools,*** such as SDKs with interface builders, widget sets, and physical building blocks (e.g. electronics);
- ***authoring tools that minimize coding*** by providing a higher-level means for authoring interactivity or for creating interactive behaviours (e.g. visual programming [5,7,15] or programming by demonstration [4,16]);
- ***high-level tools that supports debugging and understanding of the run-time system state,*** for example, Papier-Mache [6] and the Proximity Toolkit [11] provide visualizations that allows the end-developer to monitor, record, and even modify runtime information (e.g. sensor data, notifications, variables, etc.).

Yet there is another important way that toolkits vary: their intended audience. Understanding the toolkit's audience is critical, for it will influence how the toolkit will be used, what support tools should be offered, learnability, and even how the toolkit should be evaluated.

NOTABLE TOOLKITS

Our interests lie in toolkits that let the end-developer create ubicomp-style physical interfaces that: gather data from the real world (e.g. sensors); respond in software and physical objects (e.g. visualizations, motors); and support creating behaviours linking the two. We consider three toolkits within this genre that will act as running examples to discuss various toolkit/audience challenges: Phidgets [3], d.Tools [5] and the Proximity Toolkit [11].

Phidgets

Fitchett and Greenberg [3] introduced Phidgets in 2001. Phidgets comprise both hardware and software. Hardware includes USB-based circuit boards that provide different sensors and actuators. The software includes an API for interacting with each type of board. The API controls the board's components (e.g. rotating a servo motor to a particular angle) and delivers changes to sensor values as events. The software includes graphical widgets representing each board for developers to view and test the hardware counterpart.

Phidgets originated from frustrations its authors had in creating early tangible user interfaces. To build such interfaces, developers had to be knowledgeable in many areas, including circuitry, micro-programming, networking, etc. Acquiring that knowledge came at a high cost and time demand. Thus, Greenberg and Fitchett designed Phidgets with computer programmers as its audience in mind – people who do not necessarily understand electronics but are proficient in writing event-driven object-oriented software [3]. They designed Phidgets to mimic traditional UI widget programming, as it would then be easy for developers to integrate into their existing workflow.

Phidgets became a commercial product, one which is now widely recognized and used within the HCI community. Other researchers have since incorporated Phidgets into their own platforms [5,12].

d.Tools

d.Tools [5] is a high-level authoring tool, which (in part) incorporates Phidgets. A designer prototypes interactive behaviours by manipulating state-diagrams that move through different outputs based on sensor interpretations. d.Tools' audience is interaction designers – people without specialized engineering or programming knowledge who want to quickly iterate through the early designs of functional interactive objects [5]. d.Tools is widely cited in HCI. It was later

Ledo, D., Oehlberg, L. and Greenberg, S. (2017)
The Toolkit / Audience Challenge. In Proceedings of HCI.Tools, a workshop held at the ACM Conference on Human Factors in Computing Systems (ACM CHI'17). (Denver, Colorado), 4 pages, May 7. See http://hci.tools for the workshop website.

extended into Exemplar [4], which incorporated pattern recognition and programming by demonstration.

The Proximity Toolkit

The Proximity Toolkit [11] audience is highly specialized researchers investigating the design of *proxemic interactions*. Proxemic interaction imagines a world of devices and interaction behaviors that have fine-grained knowledge of nearby people and devices: how devices and people move into range, their precise distance from one another, their identity, and even their relative orientation. The toolkit encapsulates and abstracts sensor data (e.g. Vicon, Kinect), as relations between entities. Developers can focus on designing proxemic-aware applications rather than the setup and complex programming of tracking equipment and its data.

The Proximity Toolkit includes an event-driven API that informs the system of changes in proxemic values for different entities. Developers can monitor objects that move in the environment at runtime, either by showing numeric changes in variables of interest, or by interacting with a visualization showing all tracked entities and the proxemic relations between them. It also eases development by recording and storing tracked data, which can then be replayed as a simulation. The authors and their colleagues developed a large number of proxemic interaction techniques and applications [1,10] to investigate how proxemic interactions can be applied to other domains, such as advertising [17], and remote controls [8].

THE TOOLKIT/AUDIENCE CHALLENGES

Prototyping toolkits often refer to their end-developer in a range of ways: programmer, designer, developer, end-user, maker, researcher, etc. Regardless of how the expected end-developer is labelled, toolkits need to define and understand their target audience. Indeed, Olsen [14] argues for the importance of understanding situation, tasks, and user when creating a toolkit. Below are a few sample challenges that can help unpack attributes about the primary end-developer and how it relates to the toolkit.

Challenge 1. End-Developer Skills

Myers et. al. [13] argue that one aspect of evaluating a toolkit is its threshold and ceiling. *Threshold* refers to the developer effort to get started, while the *ceiling* defines how much can be done using the tools. Ideally, a toolkit would have a low threshold and high ceiling. Yet the notions of threshold and ceiling are actually relative to the skills of the end-developer.

Toolkits often extend existing programming languages, which affect the threshold for the end-developer. With Phidgets, originally built atop Visual Basic, the end-developer would have a very low ceiling only if they were proficient in Visual Basic and its interface builder. In contrast, an interaction designer with no programming background would find the threshold high, as they would have to learn to program before using Phidgets. The commercialized version of Phidgets mitigated this issue somewhat by making its API accessible to a broader audience skilled in different programming platforms: core languages (e.g., C#, Java), mobile (iOS, Android), scripting (Python), multimedia platforms (Flash), etc. d.Tools further reduce the threshold for non-programmers by providing an authoring environment that substituted programming with state-diagrams.

High ceilings also depend on the audience. Toolkits offer high ceilings through flexibility and expressiveness, but this only works when the end-developer has design skills in the area that the toolkit is trying to open. For example, the Proximity Toolkit offers a high ceiling for proxemic interaction development via: a myriad of proxemic variables; relationships between people, devices and objects; and flexible configuration of the physical sensing environment. It assumes its end-developers have knowledge in proxemic theory and how to apply it to interaction design. If the developer does not have that knowledge, the richness of the Proximity Toolkit can easily become a liability. This is especially true if the end-developer wants to take the *path of least resistance*, where the toolkit guides them to 'do the right thing, away from the wrong things' [13].

Challenge 2. End-Developer Resources

Toolkits may rely on commercial or DIY hardware. Sometimes the underlying technologies can be acquired with ease and at reasonable cost (e.g. Phidgets). However, other toolkits assume a larger infrastructure (e.g. the Proximity Toolkit requires a dedicated room and specialized hardware such as the Vicon motion tracking system). As the required resources and costs increase, the expected audience will narrow to only those very interested in the area.

Challenge 3: End-Developer Learning

Another consideration is how the end-developer will learn the toolkit.

First, end-developers need to learn what the toolkit offers over and above its base platform. For example, Phidgets and the Proximity Toolkit both offer an API to particular capabilities, and they must be learned, along with the patterns that best exploits that API. While D.Tools offers the state diagram approach, that too must be learned.

Second, end-developers also need to make sense of the overall data, automated processes, etc. as provided by the toolkit. For example, various ubicomp-oriented toolkits exploit sensor data, often delivered as low-level, frequently updated variables. Yet, learning what that sensor data means (especially if it is noisy) can be quite challenging, for that data must be related to real-world phenomena. This is partially why toolkits provide high-level tools that visualize and/or aggregate sensor data [6,11] or store information for further scrutiny [9,11]. To illustrate, the Proximity Toolkit shows a visualization of all objects in a scene, and how the proxemic variables (i.e., aggregated sensor values) track the proxemic relations between those objects. The end-developer can view the visualization to learn and understand the changes as they occur, which become references for creating the new system.

Third, research-oriented systems often assume knowledge of an underlying design paradigm. Phidgets and d.tools assume

some knowledge and experience in physical and tangible user interfaces. The Proximity Toolkit assumes some knowledge in Proxemics theory and proxemic interaction. However, the toolkits themselves do not offer easy ways to acquire that knowledge, except perhaps by referring to external resources such as publications.

Toolkits must be constructed with learnability in mind, which depends on the intended audience. They need to give the end-user an idea of what is possible, help make sense of (and debug) the data, and include resources to help new users understand the toolkit. Thus, the toolkit should offer a broad range of simple example systems, extensive documentations, repositories of examples, video tutorials, etc., as well as published papers of the design space supported by the toolkit.

SUGGESTED WORKSHOP TOPIC

Based on the above challenges, we propose the following topic for the workshop: who is the audience, and how does the toolkit design fit that audience? The prior challenges document only a few examples of concerns related to the end-developers. We expect workshop members will suggest other concerns, and elaborate on the ones mentioned here.

For example, toolkit evaluation is a great concern for many toolkit researchers, especially because submitted toolkit publications are usually accepted only if they are accompanied by a convincing evaluation of the toolkit. However, evaluation without the context of the intended audience is a somewhat pointless (and perhaps misleading) exercise. To illustrate, various toolkits are evaluated by illustrating how end-developers can quickly create prototypes within a short period of time. Yet, such an evaluation is meaningful only if the intended audience has the core skills behind the toolkit, is able to learn the toolkit quickly, and has the resources already on hand. In many toolkit papers, the audience doing the evaluation was prepared *a priori*. For example, Phidgets were evaluated by showing how undergraduate students created many Phidget prototypes quickly. Those students were already knowledgeable in Visual Basic (skills), were given a collection of Phidget hardware and cables ahead of time (resources), and were provided with lectures illustrating the tangible interface genre along with a step by step tutorial of how to use Phidgets (learnability). The Proximity Toolkit was evaluated by illustrating graduate student projects: those students were also prepared in a manner similar to the Phidgets study, and the specialized equipment required was already in place. d.Tools performed two evaluations with audiences knowledgeable in design, and were also prepared and supported. The first audience comprised participants with general design experience who were assigned particular tasks to do. The second were students in a masters-level HCI design course. In all the above cases, the context of the chosen audience approached a 'best case' for evaluation.

AUTHOR BACKGROUND AND POSITION

David Ledo is a PhD student at the University of Calgary working under supervision of Lora Oehlberg and Saul Greenberg. During his undergraduate, he took part in building toolkits (e.g. [9]), while during his masters he worked with the Proximity Toolkit, creating remote control applications [8]. Given his training and practice, he often creates libraries and wrappers for different tasks (e.g. visualization, networking). As part of teaching an undergraduate class in advanced HCI, David created a toolkit for his students to connect mobile devices and Phidgets to author new smart interactive objects. As part of his PhD topic, he works on creating prototyping tools for interaction designers to author smart interactive objects using mobile devices instead of electronic processors or components [7].

Lora Oehlberg is an Assistant Professor at the University of Calgary. Her research focuses on interactive tools and technologies that support creativity, innovation, and multi-disciplinary collaboration in domains such as interaction design, maker communities, and health care. Due to Lora's research background in product design theory and methodology, she cares about how prototyping tools and toolkits fit into real-world interaction design practice. She is interested in the intersection of interaction design and product design practice – when designers want to use prototyping tools to define the behavior and form of interactive physical objects.

Saul Greenberg is a Faculty Professor and Emeritus Professor in the Department of Computer Science at the University of Calgary. While he is a computer scientist by training, the work by Saul and his students typify the cross-discipline aspects of Human Computer Interaction, Computer Supported Cooperative Work, and Ubiquitous Computing. He and his crew are well known for their development of: toolkits for rapid prototyping of groupware and ubiquitous appliances; innovative and system designs based on observations of social phenomenon; articulation of design-oriented social science theories; and refinement of evaluation methods.

REFERENCES

1. Till Ballendat, Nicolai Marquardt, and Saul Greenberg. Proxemic interaction: designing for a proximity and orientation-aware environment. *Proc. ACM ITS 2010.*
2. Saul Greenberg. Toolkits and interface creativity. *Multimedia Tools and Applications,* 2007.
3. Saul Greenberg and Chester Fitchett. Phidgets: easy development of physical interfaces through physical widgets. *Proc. ACM UIST 2001.*
4. Björn Hartmann, Leith Abdulla, Manas Mittal, and Scott R. Klemmer. Authoring Sensor-based Interactions by Demonstration with Direct Manipulation and Pattern Recognition. Proc. ACM CHI 2007.
5. Björn Hartmann, Scott R. Klemmer, Michael Bernstein, Leith Abdulla, Brandon Burr, Avi Robinson-Mosher, and Jennifer Gee. Reflective Physical Prototyping Through Integrated Design, Test, and Analysis. *Proc. ACM UIST 2006.*
6. Scott R. Klemmer, Jack Li, James Lin, and James A. Landay. 2004. Papier-Mache: Toolkit Support for Tangible Input. *Proc. ACM CHI 2004.*

7. David Ledo, Fraser Anderson, Ryan Schmidt, Lora Oehlberg, Saul Greenberg, and Tovi Grossman. Pineal: Bringing Passive Objects to Life with Embedded Mobile Devices. *Proc. ACM CHI 2017*.
8. David Ledo, Saul Greenberg, Nicolai Marquardt, and Sebastian Boring. Proxemic-Aware Controls: Designing Remote Controls for Ubiquitous Computing Ecologies. *Proc. ACM MobileHCI 2015*.
9. David Ledo, Miguel Nacenta, Nicolai Marquardt, Sebastian Boring, and Saul Greenberg. The HapticTouch Toolkit: Enabling Exploration of Haptic Interactions. *Proc. ACM TEI 2012.*
10. Nicolai Marquardt, Till Ballendat, Sebastian Boring, Saul Greenberg, and Ken Hinckley. Gradual Engagement between Digital Devices as a Function of Proximity: From Awareness to Progressive Reveal to Information Transfer. *Proc. ACM ITS 2012*.
11. Nicolai Marquardt, Robert Diaz-Marino, Sebastian Boring, and Saul Greenberg. 2011. The proximity toolkit: prototyping proxemic interactions in ubiquitous computing ecologies. *Proc. ACM UIST 2011.*
12. Nicolai Marquardt and Saul Greenberg. Distributed Physical Interfaces with Shared Phidgets. *Proc. ACM TEI 2007.*
13. Brad Myers, Scott E. Hudson, and Randy Pausch. Past, Present, and Future of User Interface Software Tools. *ACM ToCHI* 7, 1: 3–28, 2000.
14. Dan Olsen. Evaluating user interface systems research. *Proc. ACM UIST 2007.*
15. Raf Ramakers, Kashyap Todi, and Kris Luyten. PaperPulse: An Integrated Approach for Embedding Electronics in Paper Designs. *Proc. ACM CHI 2015.*
16. Valkyrie Savage, Colin Chang, and Björn Hartmann. Sauron: Embedded Single-camera Sensing of Printed Physical User Interfaces. *Proc. ACM UIST 2013.*
17. Miaosen Wang, Sebastian Boring, and Saul Greenberg. Proxemic Peddler: A Public Advertising Display That Captures and Preserves the Attention of a Passerby. *Proc. ACM PerDis 2012.*