Empowering Designers with Creativity Support Tools

Stephen Oney
Human-Computer Interaction Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213 USA
soney@cs.cmu.edu

Abstract

When conceiving of and implementing interactive behaviors, most designers rely on professional software developers to prototype and implement their designs. They often use static drawings or animations to convey how their application should work. While these drawings are effective in conveying the look of an application, they do not effectively communicate its feel. In addition, other barriers prevent many interaction designers from taking full advantage of computational tools. We plan to address this by building a new development language and environment especially suited for creating and prototyping interactive applications. In this paper, several related studies and their implications for the design of such a language are discussed.

1. Introduction

The lack of expressiveness in nearly all modern programming languages is a significant barrier for interaction designers interested in creating novel interactive applications. Most prototyping and development tools are complex enough that when designing interactive applications, designers must rely on programmers to create prototypes and program the final application. This separation between the designer and programmer of the interactive application reduces the expression of many designers by introducing a communication barrier and reducing the amount of possible design iteration. While researchers and corporations have created several tools aimed at alleviating communication problems between designers and developers, the underlying languages and development environments used by both designers and developers often hinder their expressiveness. To address these issues, we have started the Euclase project. We aim to create a development environment especially for the creation of interactive behaviors and applications.

2. Related Work

Many past studies on designers and developers can help guide the feature-set of a tool especially for interaction designers. A few of the most relevant studies and tools are described here.

Myers et al. pointed out many of the immediate deficiencies in the tools used by designers [1]. They found that while it was easy to communicate *look*, it was much more difficult to illustrate how their application would *feel*. Most of the desired behaviors designers were interested in creating were too complex to effectively be captured by any fixed set of widgets. Designers also indicated a strong desire to be able to explore and be able to easily find and backtrack to previous designs. In addition, Brandt et al. found that most development tools do a poor job of allowing their users to easily incorporate examples into their own code [2]. They investigated the role of examples for developers, classifying the types of examples used and found that examples are crucial in the development process.

In addition to pointing out possible areas for improvement over today's tools, Park et al. have investigated the feasibility of creating a programming language that allows for natural descriptions of interactive behaviors [3]. They found that most behaviors had descriptions that were shared amongst almost all the participants. The commonalities in how certain interactive behaviors are described can help guide the design of a language especially aimed at allowing designers to prototype interactive behaviors.

3. Approach

The goal of the Euclase project is to create a language and development environment especially aimed at increasing the expressiveness of designers while creating interactive applications. To do this, I first investigated the needs of interaction designers

through a participatory design workshop, working with several other Carnegie Mellon researchers, including members of the School of Design.

In the workshop, designers and developers were paired and designers had to communicate the design of a novel control for either an online used car purchasing website or a flight booking service. Developers were paired with designers at the start of the workshop and switched partners for the second half of the workshop to be paired with different designers and different projects. This was to investigate problems designers have in communicating new and unique designs. In addition, all participants were asked to think of ways in which a tool could help improve their communication. We found that written descriptions and pictures of interactive behaviors are not sufficient to explain how an interactive behavior works for two main reasons. First, they usually require some other form of grounding or context (often in the form of an example scenario designers have in mind but don't usually write down). Second, although designers usually have an idea of what parts of their designs are uncertain and malleable, these tools are not effective at communicating the relative importance of different aspects of the design. This often leads to developers making software architectural decisions that are difficult to change if the design is modified later on.

4. New Tools

Inspired by the importance of examples to both designers and developers, I wrote FireCrystal, a Firefox add-on that helps users find code relevant to interactive behaviors [4]. With FireCrystal, developers interested in copying an interactive behavior from a foreign website can record the interaction they want to copy and then replay that interaction with FireCrystal highlighting the relevant source code and files.

While FireCrystal and other tools augment existing languages, I plan to approach many of the aforementioned obstacles by creating a new development language with features oriented towards the needs of interaction designers. Some of the crucial features of this new language are encouraging exploration, improving collaboration and communication in distributed teams, providing a syntax that is naturally suited towards interactive behaviors, and allowing for simpler debugging.

To encourage more exploration on the part of designers, the development environment could offer undo on the feature level while keeping track of what users have done across sessions. In addition, by making examples central to the language, designers could be encouraged to explore by using and customizing the prototypes of other designers. This might be done by including additional information about how the variables in examples are used, or what libraries need to be imported, in the form of additional meta-information.

Communication and collaboration possibilities amongst distributed team members could be improved by allowing for annotations on top of working prototypes, which would increase the visibility of design rationales and other notes. In addition, the same architecture that would allow the language to make better use of examples and keep track of many revisions could allow distributed teams to keep track of many different versions of a particular project and easily pick out and combine features from multiple project revisions.

5. Conclusion

Previous studies have pointed out that today's tools are not suitable for interaction designers for multiple reasons. We have built on these studies with a participatory design workshop and a preliminary tool to help designers extract example interactive behaviors. For future work, rather than building a tool on existing languages, we plan to test ways of making a new language that is more suitable for creating interactive applications and to evaluate the effectiveness of different language features. By creating a language with these features, we hope to allow more interaction designers to take advantage of the power of computational tools.

6. References

[1] B. A. Myers, S. Y. Park, Y. Nakano *et al.*, "How Designers Design and Program Interactive Behaviors," *IEEE Symposium on Visual Languages and Human-Centric Computing,*, pp. 177-184, 2008.

[2] J. Brandt, P. J. Guo, J. Lewenstein *et al.*, "Two studies of opportunistic programming: interleaving web foraging, learning, and writing code," *Proceedings of the 27th international conference on Human factors in computing systems*, 2009.

[3] S. Y. Park, B. Myers, and A. Ko, "Designers' Natural Descriptions of Interactive Behaviors," *IEEE Symposium on Visual Languages and Human-Centric Computing*, pp. 185-188, 2008.

[4] S. Oney, and B. Myers, "FireCrystal: Understanding Interactive Behaviors in Dynamic Web Pages," *Visual Languages and Human-Centric Computing*, 2009 (to appear).