

Rapid Prototyping Tools for Context-Aware Applications

Yang Li¹ & James A. Landay^{2,3}

¹Group for User interface Research
Computer Science Division
University of California, Berkeley
Berkeley, CA 94720-1776 USA
yangli@cs.berkeley.edu

²Intel Research Seattle
1100 NE 45th Street
Suite 600
Seattle, WA 98105
james.a.landay@intel.com

³DUB Group
Computer Science and Engineering
University of Washington
Seattle, WA 98105-4615 USA
landay@cs.washington.edu

ABSTRACT

Context-aware applications are one of the most important forms of next generation interactive systems. We discuss new attributes exhibited by context-aware interactions and introduce our experience with developing a tool for prototyping location-enhanced applications, which are a useful subset of context-aware applications. Based on this, we discuss what challenges we face in developing a more general context-aware application design tool and how these challenges can be addressed. In addition, we briefly describe our ongoing work on building such a tool for prototyping general context-aware applications based on human activities.

INTRODUCTION

Ubiquitous computing will tremendously shift the paradigm of human computer interaction. Among many features exhibited by these emerging user interfaces, context-awareness of interactions is a unique perspective to explore the next generation of user interfaces, which essentially broadens the bandwidth of HCI within a context-rich ubiquitous environment.

User interface prototyping tools have gained great success for designing traditional user interfaces. In particular, informal prototyping tools allow designers to quickly prototype, test and refine an idea in the early stages of design [3, 5].

As ubiquitous computing develops, prototyping tools for context-aware applications will be in demand. These tools can lower the barrier to entry so that interaction designers can participate in the development of applications that currently requires a high-level of technical expertise. Tools can also help produce more usable context-aware applications in an efficient way. More importantly, these prototyping tools can help shape the future of ubiquitous computing and eventually accelerate the development of next generation user interfaces.

ATTRIBUTES OF CONTEXT-AWARE INTERACTIONS

To build a useful tool, it is important to identify the attributes of context-aware interactions. Here we describe several major distinctions from traditional GUI interactions. In addition to explicit interactions such as selecting an object via a mouse click, context-aware interactions allow

implicit interactions based on context input, e.g., automatically choosing (or showing) nearby objects based on a user's current location. The implicit interactions allow a semantics-rich input vocabulary (literally, any information contextually relevant in a situation) and a more dynamic interaction flow.

Context acquisition depends heavily on inferences based on sensed data, e.g., GPS or accelerometer readings. Consequently, context input is inherently *ambiguous*, e.g., a reported location is not necessarily the actual location of a user's device.

Contextual input is often extracted using continuous observations of a series of events over a period of time, i.e., analyzing human activities based on a sequence of GPS readings. Incremental feedback based on this continuous input is often required, e.g., continuous updating of a user's location on a map. As such, it is natural to consider this type of interaction as continuous¹.

Contextual information is the glue that integrates the physical and electronic worlds. It is common that *multiple users*, e.g., a lecturer and a group of students in a classroom, and *multiple modalities*, e.g., a lecturer gesturing while speaking, are involved in a context-aware interaction. Thus, identities of entities are an important piece of contextual information and it is more efficient to integrate multiple modalities with the understanding of the context.

PROTOTYPING LOCATION-ENHANCED APPLICATIONS

We started our exploration of tools for prototyping context aware applications by narrowing down our focus to location contexts. This research resulted in Topiary [4], the first tool for prototyping location-enhanced applications. Location-enhanced applications adapt their behavior based on the locations of people, places and things. This is the most widely adopted type of context-aware system today. Topiary provides a set of high level, concrete abstractions for prototyping: maps, scenarios and storyboards. As a result, the tool can be grasped by designers without any programming or hardware background. Topiary employs

¹ An interaction can be considered either continuous or discrete at different levels of granularity.

Wizard of Oz techniques to allow for easily testing a design in the field.

RESEARCH CHALLENGES AND POSSIBLE SOLUTIONS

Based on our experience in developing and evaluating Topiary, we propose several key challenges for developing context-aware applications and discuss possible solutions.

Modeling the New Input Vocabulary of Contexts

The use of contextual input results in a new input vocabulary that has not been deeply studied and well classified as compared to mouse and keyboard events in traditional user interfaces. Consequently, designers have to explore a much larger input space and specify desired contexts while prototyping a context-aware application. A design tool should facilitate this task by providing high level abstractions. In Topiary, we employed a map abstraction to represent spatial relations of entities and allowed designers to capture location contexts of interest by demonstrating scenarios.

Modeling Implicit and Continuous Interactions

Storyboards have been employed by many tools for prototyping traditional interactions that are explicit and discrete [3, 5]. In designing Topiary, we vastly expanded the storyboarding visual language by introducing the notion of implicit links. These are storyboard transitions that are activated via implicit location input. We also introduced explicit links conditioned on location contexts.

In addition, Topiary employed context components to partially address the issue of continuous interactions. For example, the active map component continuously updates the user's location on a map. However, is storyboarding, which is inherently discrete, sufficient for modeling continuous context-aware interactions? We are studying this issue in our current work.

Modeling Ambiguity

Explicit support for modeling ambiguity is useful when making a design, as this can help uncover related usability issues in the early stages of design. Suede [2] allows designers to specify the accuracy of speech recognition. Prototyping tools for context-aware applications should have a similar mechanism, though the error model could be richer than the single threshold used in Suede.

Designing Wizard's Interfaces for Wizard of Oz Testing

Because testing with real sensors can be very expensive, Wizard of Oz techniques are a promising way of testing context-aware applications, as demonstrated by Topiary. As the target setting of a context-aware application is often dynamic, e.g., in the field, it is a challenging issue to *design an effective wizard interface*. A design tool could automatically generate wizard interfaces. Based on our experience with Topiary, we found that there is often a *tradeoff* in the appropriate task allocation between designers and wizards. For example, if a tool requires that

more detailed and complete behaviors of a prototype be specified at design time, then there will be less work for wizards at test time and vice versa.

In addition to some common concerns of Wizard of Oz testing, such as keeping the wizard interface simple or minimizing the disturbance of the wizard on test participants, it is important for us, the tool builders, to design a wizard input language that best approximates the direct observations the wizard makes in the target setting. This can help lower the cognitive load of the wizard as they manually translate their observations into updates of the prototype's state. For example, a wizard need only update a user's location on a map in Topiary rather than inferring high level information about the current scenario. In this case, the direct observation of a wizard is a user's movement and that is exactly the state that they update.

ACTIVITY-BASED DESIGN TOOLS

We are currently designing an activity-based design tool for general context-aware applications. Human activities that involve people, tools and objects give both a more general and a higher level interaction vocabulary than location contexts. Activities can also give a more complete picture of contextual information.

Two Foundations Building Such a Tool

Activity Theory [1] provides a broad conceptual framework for describing the structure, development and context of computer-mediated activities, i.e., HCI. An activity is undertaken by a *human agent* who is motivated toward the *solution of a problem or purpose*, and mediated by *tools in collaboration with others*. It gives us theoretical guidance as to what basic elements should be included in a design tool and how these elements can interact with each other.

While Activity Theory gives us a conceptual foundation, *Temporal Probabilistic Models* [7] give us a computational basis and structural view of activities for developing such a tool. An activity is often inferred from a sequence of low level information such as interactions with objects [6]. The model not only contributes to the inference of activities but also implies a design metaphor based on states of an activity and transitions between these states.

Basic Concepts and Proposed Features

In our current design sketch, we have identified several basic concepts (or elements) for the tool. An *activity* includes a set of stages and possibly temporal transitions between stages. A *stage* has a set of actions being carried out simultaneously in the sense that they belong to the same stage. An *action* involves a user and an artifact. An *artifact* can be a physical tool in the physical world, e.g., a cup, or a software tool. Based on the structure of activities, a designer's task is to design appropriate user interfaces so that a computer tool can facilitate the execution of activities. Two typical cases are automation of activities and automatic adaptation of user interfaces. For example, in the first case a cooking assistant automatically shows

different ingredients necessary at different stages of following a recipe. In the second case, a cell phone interface might automatically switch to silent mode when in a meeting.

Our proposed tool will provide access to an existing activity database [6] so that designers can design based on this large set of activities. The tool should also allow designers to create new activities. Finally, designers should also be able to quickly test a design using Wizard of Oz techniques.

ACKNOWLEDGMENTS

This work has been supported by NSF under Grant No. IIS-0205644.

REFERENCES

1. Bertelsen, O.W. and Bødker, S., eds. *Activity Theory. HCI Models, Theories, and Frameworks: Toward an Multidisciplinary Science*, ed. J. Carroll. 2003, Morgan Kaufman. 291-324.
2. Klemmer, S.R., et al., SUEDE: A Wizard of Oz Prototyping Tool for Speech User Interfaces. *UIST 2000, CHI Letters*, 2000. **2**(2): pp. 1-10.
3. Landay, J.A. and Myers, B.A., Sketching Interfaces: Toward More Human Interface Design. *IEEE Computer*. **34**(3): pp. 56-64.
4. Li, Y., Hong, J.I., and Landay, J.A., Topiary: A Tool for Prototyping Location-Enhanced Applications. *UIST 2004, CHI Letters*, 2004. **6**(2): pp. 217-226.
5. Newman, M.W., et al., DENIM: An Informal Web Site Design Tool Inspired by Observations of Practice. *Human-Computer Interaction*, 2003. **18**(3): pp. 259–324.
6. Philipose, M., et al., Inferring Activities from Interactions with Objects. *IEEE Pervasive Computing*, 2004. **3**(4): pp. 50-57.
7. Russell, S. and Norvig, P., *Probabilistic Reasoning over Time*. 2 ed. Artificial Intelligence: A Modern Approach. 2003: Prentice Hall. 537-583.