Bringing the Field into the Lab: Supporting Capture and RePlay of Contextual Data for Design

Mark W. Newman[1], Mark S. Ackerman[1,2], Jungwoo Kim[2],
Atul Prakash[2], Zhenan Hong[1], Jacob Mandel[1], Tao Dong[1]
[1]School of Information, [2]Department of Electrical Engineering and Computer Science
University of Michigan, Ann Arbor, MI 48109
{mwnewman, ackerm, jungwook, aprakash, guoerhzn, jacobman, dongtao}@umich.edu

ABSTRACT
When designing context-aware applications, it is difficult to for designers in the studio or lab to envision the contextual conditions that will be encountered at runtime. Designers need a tool that can create/re-create naturalistic contextual states and transitions, so that they can evaluate an application under expected contexts. We have designed and developed RePlay: a system for capturing and playing back sensor traces representing scenarios of use. RePlay contributes to research on ubicomp design tools by embodying a structured approach to the capture and playback of contextual data. In particular, RePlay supports: capturing naturalistic data through *Capture Probes*, encapsulating scenarios of use through *Episodes*, and supporting exploratory manipulation of scenarios through *Transforms*. Our experiences using RePlay in internal design projects illustrate its potential benefits for ubicomp design.

ACM Classification: H5.2: User Interfaces. - Prototyping.

General terms: Design, Human Factors

Keywords: context-aware, design tools, data capture.

INTRODUCTION
The changing contexts for ubicomp applications make it difficult for designers to understand how design and implementation decisions will impact the user experience. As a result, designers often see it as necessary to build and deploy high fidelity prototypes early in the process in order to obtain insights into the user experience. Building such high-fidelity prototypes is generally a cumbersome and expensive task, as it involves deploying sensors and devices in the target environment in addition to the application being designed. If designers can obtain early insights into how applications will behave under usage conditions, their designs can be modified to deliver better a user experience while the changes are still relatively easy to make.

A number of approaches have been explored to address this problem and to make it easier for designers to create and test designs both inside and outside the development lab. A fruitful approach that has been explored in multiple projects is combining rapid prototype construction with support for Wizard of Oz (WOz) testing (e.g., [6, 7]) in order to accelerate the design-build-test cycle. One significant advantage of WOz is that it allows designers to *bring the lab into the field* by enabling them to observe informative usage experiences with partially formed prototypes. A complementary but somewhat less explored approach that has been to provide support for simulating ubicomp environments (e.g., [2]). The simulation approach seeks primarily to *bring the field into the lab* by representing important aspects of the intended user experience such as the physical environment into which deployment is expected, the devices that are expected to be used, and the behavior of other users. Both approaches are needed. Lo-fi prototyping and WOz support enable iterative design in the large by allowing design ideas to be tested by users in realistic settings. Simulation approaches enable iterative design in the small by supporting reflective prototyping [5], i.e., the practice of rapidly exploring and reflecting upon multiple design variations during prototype construction.

In this paper, we contribute an alternative approach to "bringing the field into the lab" that avoids the key shortcomings of simulation approaches—namely that building and configuring models of user behavior is difficult and that the resulting models are often poor approximations of real behavior. Our approach allows a designer to:

- Capture naturalistic user behavior through the use of *Capture Probes,*
- Construct collections of user traces called *Episodes* representing usage scenarios, and
- Create ad hoc permutations of traces and Episodes through the use of *Transforms* to allow the designer to explore different conditions that will be faced by the application.

RELATED WORK
While a number of systems have supported capture and playback of events for conducting user tests (e.g., [9]) or analyzing usage in the field (e.g., [8]), a smaller subset of systems have supported the playback of captured and/or hand-crafted event streams for building context-aware prototypes. Topiary [6], for example, allows designers to define context-based triggers for UI transitions that can be

UIST'10, October 3–6, 2009, New York, New York, USA.

manipulated by designers (or "wizards") during prototyping and field testing. *a CAPpella* [4] supports programming by demonstration of context-triggered events by capturing sensor traces and allowing end-users to define events of interest that should be recognized by the system. DART [7] supports the capture of head tracking and location sensor data from field tests for use in revising prototypes of Augmented Reality applications.

Our work builds upon these previous efforts by providing more comprehensive and structured support for capture and playback of sensor data in ubicomp design. Namely, we provide support for initial data capture, synthesis of disparate traces into coherent representations of human activity, and manipulation of those representations to explore various contextual conditions that an application must handle.

THE REPLAY SYSTEM

RePlay forms one part of a ubicomp designer's toolchain. We anticipate that RePlay would sit alongside or be integrated into low-fi prototyping tools such as Topiary [6] and/or development environments such as Eclipse (www.eclipse.org/). In this paper we ignore the provenance of the prototypes.

RePlay Implementation and Underpinnings

RePlay is currently built in Java and interacts with the Whereabouts infrastructure [1]. For the purposes of RePlay, Whereabouts is an XML-based tuplespace that allows any sensor, service, or client to publish and/or subscribe to arbitrary messages. In this regard, Whereabouts is similar to other ubicomp event-based infrastructures. RePlay draws event tuples from a database of stored traces and posts them to Whereabouts. Clients, including our prototypes receive these tuples just as they would live sensor feeds, transparently unaware that they are responding to replayed data.

Capturing Sensor Traces with "Probes"

In RePlay, *Capture Probes* are portable and/or embeddable sensors that can be used by designers to capture sensor data relating to situations that need to be supported. For example, a designer wishing to provide location-based support for users moving through an urban environment might capture the natural movements of representative users over several days in order to guide the system design using a mobile Probe. A designer wishing to add responsive ambient media into a public lounge might embed a Probe to capture the comings and goings of people within that lounge over some period of time. Probes are designed to be deployed into environments and situations of interest without requiring supervision, and then collected by the designer for import into RePlay. To date, we have implemented two probes: an Android-based *GPS Location Probe* that collects a user's location across several days, and a Linux-based *Bluetooth Proximity Probe* that records the signatures of users' Bluetooth-enabled devices (mobile phones in particular) in a fixed environment.

Synthesizing Collections of Traces into "Episodes"

Finite, time-bounded sensor trace sequences, extracted from the traces collected by Capture Probes, are referred to as *Clips*. The *Clip Library* is a database of all clips that are available to a designer, supporting queries across metadata including data source, type, creation date, and duration.

RePlay allows the designer to organize Clips into synchronized groups of clips called *Episodes* that represent coherent sequences of human activity. A key motivation behind the inclusion of Episodes is to provide support for Scenario-Based Design [3]. Interaction designers often make use of scenarios as a way to encapsulate user behaviors with respect to a system for the purpose of guiding design. They are most often represented as textual narratives that tell a story about user behavior, and are used by designers to foster awareness of the activities the system needs to support and the context in which those activities are likely to be enacted. From this viewpoint, we view Episodes as executable scenarios—i.e., as a way for designers to construct coherent groupings of sensed behavior that can be played back in concert. Thus designers can test a prototype against traces of real user behavior to ensure that the system will work well in expected cases.

Episodes allow Clips to be combined together in flexible ways. For example, the designer of a location-based social application may wish to create a scenario in which two friends pass near each other within a short period of time even though the collected data may not contain such a segment. For such cases, the Episode abstraction allows the combination of Clips from disparate time periods, and allows the designer to control the relative time offsets of each Clip within the Episode.

Permuting Captured Traces with "Transforms"

To create sufficient data and to fully examine a design, designers will need to see how their applications respond to different permutations of captured sensor data. RePlay supports this need by allowing designers to apply one or more *Transforms* to each clip. A transform is a unit of processing that operates on a clip to alter its data in particular ways. For example, a sequence of data that was collected from one source (e.g., a user's smartphone) could be transformed to appear as though it is data from another source (e.g., a different user's smartphone). Since the association between Transforms and Clips is maintained by the Episode, Transforms can be non-destructively applied. Clips can participate in multiple Episodes with different parameters.

RePlay is architected to support a wide range of transforms. Three examples can serve as illustrations of important classes of transforms. The *Identity Transform* changes the user associated with a particular transform, allowing the designer to experiment with different user views, preferences, and settings within an application. The *Dwell Transform* introduces a delay within in a clip to simulate a period of stability within a clip that lasts longer than what was actually observed. For example, a users' presence in a particular location can be artificially lengthened to simulate a

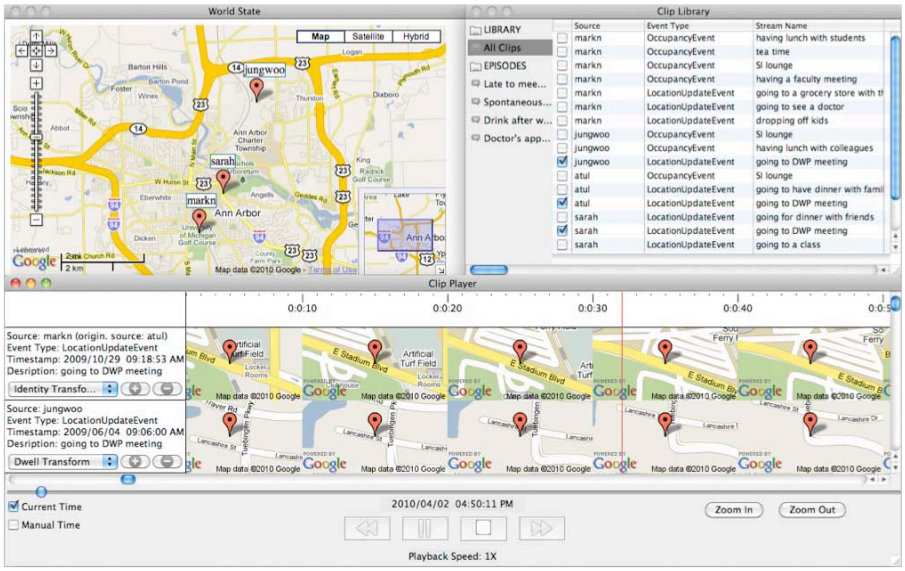

Figure 1. The RePlay user interface consists of three main components. Clockwise from the top left, they are: the World State window, which shows the current states of all sensed data from the clips in the Clip Player; the Clip Library, which provides access to all previously captured clips; and the Clip Player, which shows all currently active clips along with controls for starting, stopping, and changing playback speed of the current session.

situation in which a user "meets up with" another user. The *GPS Noise Transform* operates on Location data and applies a designer-specified degree of random permutation of the GPS-reported location (realistic GPS noise models are currently under development). This allows designers to see how their location-based application would respond to data that is noisier than the data actually collected.

Additionally, RePlay supports *Active Transforms* that interpret the contents of a stream and generate new events, effectively supporting the prototyping of new services within RePlay. As an example, we have implemented the *Colocation Transform* that monitors the location of two or more users' location streams and reports when they are within a specified distance of one another.

The RePlay User Interface

The RePlay UI is shown in Figure 1. Like *a CAPpella* [4], it draws inspiration from multi-track media editors such as Final Cut Pro (www.apple.com/finalcutstudio/), which share with RePlay the requirement that a user be able to select, control, and preview synchronized sequences of events.

The RePlay UI consists of three elements. The *Clip Player* window shows a multi-track timeline representation of each currently active clip's data along with an indicator of the current session's playback status. It also provides controls to allow the designer to play, pause, speed up, slow down, and reverse the playback session. In addition, the designer can apply, remove, and configure Transforms by selecting them from a list under the name of the clip. The *Clip Library* window provides access to a library of clips or other sources that can be selected for inclusion in the current playback session by moving the clip to the Clip Player. Clips can be added or removed from Episodes via drag and drop within this window. The *World State* window shows a representation of the playback's current state that is synchronized with the time index in the Clip Player. RePlay provides a plug-in architecture for rendering data streams in both the World State and Clip Player windows. At present, we have developed renderers for location data (Figure 1), single-valued numerical data, booleans, and strings.

EXPERIENCES WITH REPLAY

We have used RePlay for our own internal application design projects, and in this section we present one such experience. This account is intended to elaborate our arguments that RePlay can provide benefits during design, not to claim empirical evidence of RePlay's acceptance by practicing designers.

Improving MFinder

MFinder [1] is a mobile application that allows users to share their current location with other "friends" and to see those friends' locations on their mobile device, similar to popular tracking-based "friend finder" mobile apps like Loopt (loopt.com/). For the purposes of our exploration of RePlay as a design tool, we examined how to add features to MFinder. Prior to our design exercise, MFinder existed in a rudimentary form, and RePlay played a valuable role in the development of additional features improving MFinder's user experience. We provide a discussion of one such feature as an illustration of RePlay's usefulness.

Designing MFinder's "Estimated Time of Arrival" Feature

One of our group members is sometimes late for meetings. It would be valuable for the other meeting attendees to know if the tardy member is well on his way, is just starting out, or is in a location that would indicate he has messed up his calendar. We have come to refer to this situation as the "Where is Mark?" scenario, and we decided to use RePlay

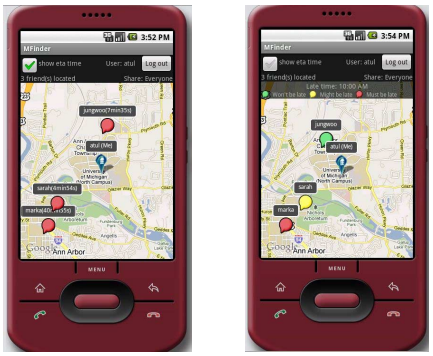

Figure 2. RePlay allowed us to experiment with two different presentations for the ETA feature in MFinder: adding time estimates to the user's labels and encoding the estimates as icon color.

to explore ways that MFinder could help other group members understand the tardy member's status and the likely time of his arrival, if any, at the meeting location.

One feature we decided to explore is the *Estimated Time of Arrival* (ETA) attribute that would indicate, for all participants in a meeting, the approximate time one should expect them to arrive. There are of course many design decisions to be made with respect to this feature, including algorithms for estimating the actual time of arrival, but due to space we will focus on one specific decision for which RePlay was particularly helpful. Using RePlay, we explored two options for the visual display of ETAs (shown in Figure 2): first, the addition of the best-guess ETA to the label associated with each person's map icon and second, the encoding of ETAs into the icon color.

Our process for using RePlay in the design of the ETA feature consisted of the following steps: First, we captured several days of location traces for all of our group using the GPS Location Probe. We then constructed an Episode representing several group members converging on the meeting location, including a Clip showing "Mark" arriving 20 minutes late due to a diversion on the way to work. Next, after creating variants of MFinder for the above-mentioned designs for displaying ETA information, we played the captured data through both versions of MFinder to examine the issues in using one approach versus the other.

As we played the captured scenario data through the client variants, we manipulated the Clip data by applying GPS Noise and Dwell Transforms in order to see how poor GPS signal would impact the ETA estimates and resulting user experience, and how commuter delays or other unexpected issues might impact users' ability to make sense of a tardy member's progress given each UI design. In addition, we tested the UI with different numbers of user traces by adding and removing Clips from the "Where is Mark" Episode.

From this process, we determined that the color-coding approach would most likely yield the best user experience for two reasons: (1) adding additional text to each icon's label becomes unwieldy with more than 5 and (2) color coding did not give an artificial sense of precision that could be misleading for viewers. In addition, we noted several situations that were problematic for the UI that would not have been noticed without RePlay, such as wildly inaccurate ETA predictions when users pause in transit.

In our design of the ETA feature, RePlay was helpful in rapidly exploring our design options. The data representing meeting arrivals was easily collected by distributing smartphones to group members. The "Where is Mark" scenario was able to be represented by an Episode, and variants of the Episode were easily constructed in order to explore variants of the original scenario. Transforms, especially the Dwell Transform, were valuable in identifying problems with the UI that require followup.

CONCLUSION

The key advantage of RePlay is that it allows designers to address design issues and explore application behavior during the design process without resorting to full-fledged field deployments. RePlay allows designers to capture naturalistic sensor traces in order to encapsulate expected contextual states that an application will encounter. It also allows the designer to manipulate those encapsulated Episodes through the use of Transforms. Our initial experiences with RePlay indicate that our approach offers promise toward our goal of providing a tool to allow designers of context-aware applications to explore design options more rapidly than they are currently able to do. Future work will include scaling RePlay to support working with larger libraries of captured clips and extending it to work with additional types of sensors and applications.

ACKNOWLEDGMENTS

We thank Zhe (Jo) Pu, Sarah Qidwai, and Ben Congleton. This work was sponsored in part by Intel Research and the National Science Foundation (0705672 and 0905460).

REFERENCES

1. Ackerman, M.S., et al. Simplifying user-controlled privacy policies (Work in Progress). *IEEE Pervasive Computing 8*, 4 (2009), 28-32.
2. Barton, J.J. and Vijayaraghavan, V.: UBIWISE, a simulator for ubiquitous computing systems design. Hewlett-Packard Labs Technical Report HPL-2003-93, 2003.
3. Carroll, J.M. *Scenario-Based Design.* John Wiley & Sons, 1995.
4. Dey, A.K., et al. a CAPpella: programming by demonstration of context-aware applications. *CHI 2004*, 33-40.
5. Hartmann, B., et al. Reflective physical prototyping through integrated design, test, and analysis. *UIST 2006*, 299-308.
6. Li, Y., Hong, J.I. and Landay, J.A. Topiary: a tool for prototyping location-enhanced applications. *UIST 2004*, 217-226.
7. MacIntyre, B., et al. DART: a toolkit for rapid design exploration of augmented reality experiences. *UIST 2004*, 197-206.
8. Morrison, A., Tennent, P. and Chalmers, M. Coordinated Visualisation of Video and System Log Data. *IEEE CVE 2006*, 91-102.
9. Welbourne, E., et al. Specification and Verification of Complex Location Events with Panoramic. *Pervasive 2010*, 57-75.