Puppet Prototyping: Wizard of Oz Support throughout an Iterative Design Process

Steven Dow[1], Blair MacIntyre[1], Jaemin Lee[1], Christopher Oezbek[1], Jay David Bolter[2], Maribeth Gandy[3]
College of Computing[1], School of Literature, Communication and Culture[2],
Interactive Media Technology Center[3]
GVU Center, Georgia Institute of Technology
Atlanta, GA 30332, USA
{steven, blair, jaemn}@cc.gatech.edu, oezbek@inf.fu-berlin.de, jay.bolter@lcc.gatech.edu,
maribeth@imtc.gatech.edu

Abstract

Although the Wizard of Oz (WOz) method for simulating system components is commonly used for evaluation in HCI, researchers and designers have only started to unlock the potential of this technique. In this paper, we review the Wizard of Oz method and highlight its usefulness throughout the evolution of a user interface or system. We point toward a design space for WOz simulation, where the Wizard takes on different roles (such as a controller, a moderator or a supervisor). We argue that explicitly supporting WOz prototyping in pervasive computing infrastructures will improve the usefulness of this method. We describe the WOz simulation features of DART, a mixed reality design and prototyping environment, including a lightweight method for automatically generating Wizard of Oz interfaces and high-level support for visualizing evaluation results. Finally, we use a location-aware audio experience in the historic Oakland Cemetery in Atlanta as a case study to illustrate how a wizard operator can play distinct roles throughout an iterative design process.

Keywords

Wizard of Oz, Prototyping, Design Process, Ubiquitous Computing, Mixed Reality, Audio Tours, HCI Methods

Introduction

The Wizard of Oz (WOz) simulation method is a prototyping approach widely used among researchers and professionals in human computer interaction (HCI). A "wizard" operator generally plays some role in a work-in-progress computer system, such as simulating sensor data, contextual information, or system intelligence; we use the complimentary word "puppet" to refer to the mocked-up user interface controlled by the wizard. Although the WOz method can play a role in traditional desktop interaction, it is particularly useful in exploring pervasive, ubiquitous or mixed reality systems that combine complex sensing (e.g., computer vision, activity recognition, voice recognition) and "intelligent" control logic. With such non-traditional interfaces, the vast design space provides many possibilities for user interaction through one or more modalities and often requires challenging hardware and software implementations. The WOz method helps designers avoid getting locked into a particular design or from working under an incorrect set of assumptions about the users' preferred mode of interaction, because it enables design exploration and evaluation before investing the considerable development time needed to build a complete prototype.

As with other "throw away" tools, most WOz interfaces are not conceived to evolve with the system. As a result, the WOz method tends to be used once (or perhaps twice) during the evolution of a system, in sharp contrast to other evaluation methods that tend to be used repeatedly as a system evolves. This is unfortunate, as WOz studies are a useful tool for evaluating partially complete systems and to advance the design of the underlying technology. We believe that one reason for the limited use of the WOz method is the effort required to engineer a successful WOz interface and integrate it with a partially complete system. We address this problem in DART (the Designer's Augmented Reality Toolkit, our design environment for augmented and mixed reality systems [1]) by providing explicit WOz support within the design environment. In particular, we explicitly support the co-evolution of WOz interfaces with the incomplete, yet evolving, interactive system. By reducing the barriers to creating and reusing WOz interfaces throughout the design cycle, designers are more likely to utilize this evaluation method.

Based on our experience with WOz simulation in mixed reality applications and our survey of the uses of WOz in the literature, we believe there are opportunities to better exploit the WOz strategy. In our previous work, we argue that this method is valuable throughout the design process and that design environments can provide more explicit support [2]. We expand on our initial work introducing a preliminary framework for wizard roles, providing more detail about our wizard tool support in DART, and supporting post-WOz data analysis. In this paper, we argue that if WOz tools become a fluid component of a natural design process, wizards can take on a variety of roles with differing levels of responsibility for facilitating the interaction between the computing system and the user. We briefly discuss DART (the design environment) and the event-based architecture that supports using WOz simulation at various levels of abstraction. We present a case study of "The Voices of Oakland" and discuss the transition between a series of WOz interfaces enabled by the explicit support for WOz simulation in DART. We expose three distinct wizard roles employed throughout several iterations of the design and demonstrate how visualizing WOz study data can help determine a design project's future directions.

Iterative Design and the Roles for Wizard of Oz

Interactive system design typically involves an iterative process of brainstorming, prototyping, developing, user testing, and evaluation. This is not a clear-cut process, and often iterates through many cycles before reaching a final system. WOz techniques have been used in a variety of ways, depending on the technologies used and the specific goals of the project (see the Survey in the next section).

Figure 1 is a stylized illustration of how the WOz method can be used throughout the design process, as the system evolves, to close the gap between the current system capabilities and the imagined system.   (In practice, the development cycle is much more complicated, including the likelihood that the vision for the system changes at various points, altering the technology development curve.) During any WOz-based evaluation, the wizard essentially becomes a crutch for simulating the envisioned interface and interactions before the system works entirely. As technology development progresses, the gap filled by the wizard shrinks.

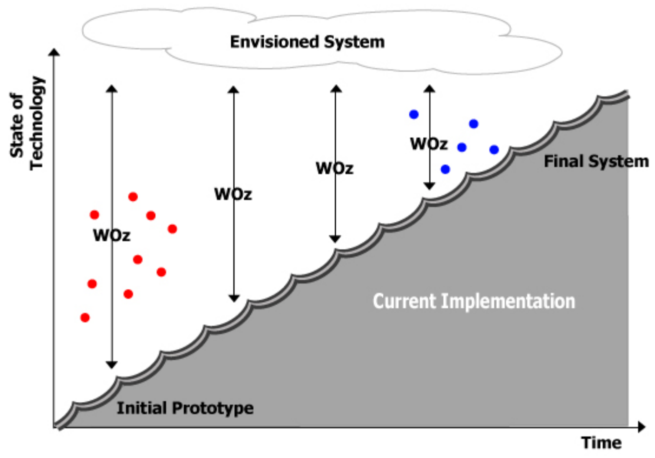


Figure 1: One general view of design evolution and the role WOz simulation plays in facilitating the evaluation of the envisioned experience. The WOz method has traditionally been used early to simulate undeveloped technology (examples indicated as red dots) or near the end where the wizard can supervise a full implementation or simulate a small piece of the technology (blue dots). There are opportunities for WOz simulation in the middle ground as a means for transitioning to a final design.

Traditionally, a wizard operator plays one of two roles: a *controller* wizard fully simulates some system component (be it sensor simulation or system intelligence) that has not been built, and a *supervisor* wizard oversees a working system and, if necessary, overrides decisions made by the system or user. The blue and red dots in Figure 1 represent examples of where WOz simulation has been used in the published literature.

A less common role, which we have used in our own work, is that of *moderator*. The moderator role lies between the controller and supervisor; when a technology or system component is working but not yet trusted, instead of hooking it into the system, a moderator wizard can be used to observe the output of this component and send this as input to the rest of the system. However, because the moderator can override the system component or sensor before the output reaches the rest of the system, a moderated evaluation can still provide the user with the envisioned user experience. This allows useful evaluation feedback to be obtained, while collecting a rich set of meta-data for evaluating the development of the system component or sensor in question (i.e. assuming the evaluation session is being logged, and the wizard is implicitly tagging the log when she either takes the action suggested by the component or overrides it). By using the moderator role to ensure the user receives the envisioned experience, despite unexpected behaviors by the underlying system, much more realistic evaluation can be performed than if the system behaved erratically or failed completely. By seriously analyzing what was happening when the wizard diverged from system decisions, the evaluation may also uncover false assumptions in the design. Conducting WOz studies at intermediate stages of system development helps refine the user interaction while informing the technology development, especially if rich user interaction and wizard input data is collected during the moderated experience.

Although it is appealing to proclaim a natural transition from controller to moderator to supervisor with diminishing cognitive load on the wizard, the design process is not this straightforward. Nor do these descriptions represent an exhaustive list of specific roles, but are rather intended to reveal the spectrum of possibilities for WOz prototyping. We hope that by making it easy to use WOz simulation throughout the design cycle, we might develop a more comprehensive framework of possible wizard roles. Therefore, we advocate including facilities within pervasive computing prototyping environments to enable designers to leverage different WOz roles and evaluation strategies as the need arises.

Survey of Wizard of Oz in HCI

Although this is not by any means an exhaustive survey of WOz, we hope to provide an overview of examples and reflections on this technique. The most common role for the Wizard of Oz in human computer interface development places responsibility on the wizard to fully control a missing piece of the technology infrastructure (from simulating the entire system to simulating one sensor). Used early in design as a lightweight prototyping method, the WOz method presents users with rough sketches of interface ideas, even when it is unclear what the underlying technology should be. In the “interruptibility” study, Hudson et al. wanted to understand users’ willingness to be interrupted in an office environment [3]. Their WOz study helped them identify what type of sensors would be appropriate in the space. Voida et al. utilized WOz to study basic interaction techniques for a projector/camera-based augmented reality environment [4]. They wanted to understand user behavior and preferred interaction unconstrained by technology-imposed limitations, such as special gloves or highly constrained movements to aid a computer vision subsystem.

Further in the design process, the appropriate technologies may be identified, but still be too difficult to implement, especially for testing speculative user interface issues. In Topiary, the wizard plays the role of a location sensor (e.g., the global positioning system) during the design of location-based applications [5]. In Lyons’ evaluation of dual-purpose speech, the WOz method enabled him to explore a larger portion of the design space by analyzing unrestricted speech input for novices [6]. Tran et al. use a wizard to simulate vision technology during the development of the Cook’s Collage, a memory aid for elders in a kitchen environment [7]. Using Woz, prototypes like Cook’s Collage can mature into sophisticated applications that help researchers test theories of interaction without spending time over-engineering a complex underlying system that may not be needed in the final product.

Even in finished applications where the system does most of the work, a wizard can play a variety of roles. In Alice’s Adventures in New Media, a wizard operator simulates a gesture recognizer as input to a sophisticated, agent-based narrative engine as part of an augmented reality experience [8]. All of theses example position the wizard as a controller, whether it is early or late in the design process. In the mixed-reality performance Desert Rain, the wizard plays the role of supervisor, helping participants through the experience on a case-by-case basis [9]. In a final experience a wizard can play a dedicated role, add intelligence beyond the current possibilities of computing, or simply monitor the experience to help when problems arise, much like an amusement park ride operator.

Other authors have pointed out general considerations for WOz studies. An early reflection by Dahlback et al. revealed a need for careful design of WOz simulations in natural-language dialogue systems (although their observations are generally applicable); researchers must pay attention to non-simulated parts of the system, the constraints of the user task, and the guidelines for the wizard [10]. As Molin indicates, user awareness of the wizard and incorrect wizard behavior can taint the evaluation and compromise the results [11]. Maulsby et al. reflect on their use of WOz stating, among other things, that a designer benefits greatly by becoming the wizard and formal models for system behavior are necessary for keeping the simulation honest [12].

In the SUEDE system for prototyping and evaluating speech user interfaces, Klemmer et al. reveal the need for simulating system error to realistically evaluate user performance during WOz studies [13]. The work on SUEDE supports our argument that WOz studies should simulate the envisioned interaction, even when generating the envisioned system requires modification to the wizard input. These considerations all point to the need for careful and appropriate experimental design in WOz simulations. WOz user studies should be oriented to answer existing design questions, and the wizard's role in the experimental design should be well-defined and consistent.

Several research projects place emphasis on WOz simulation in development environments, but generally limit the design space to one interface domain (such as speech recognition in the SUEDE tool and location simulation in Topiary [12, 5]). The Neimo project developed a WOz testing environment for the study of multimodal systems where one or multiple wizard operators can supplement missing system functions [14]; they place emphasis on the physical arrangement in the testing environment. Our work on the Designer's Augmented Reality Toolkit (DART) focuses on the rapid prototyping of more general applications in mixed reality and ubiquitous computing [1]. While DART is not appropriate for all pervasive computing systems, it illustrates how a judicious choice of programming model can enable WOz tools to be integrated into a general-purpose design environment.

Tools to Support the Wizard of Oz Technique

To describe our support for designing and analyzing Wizard of Oz studies, we introduce the design environment we built for prototyping mixed reality and ubiquitous computing systems. We explain the event-based model and networking infrastructure used in DART to support WOz. We then provide technical details about the high-level tools integrated into the prototyping environment to support rapid development of wizard interfaces and visualization of wizard study results.

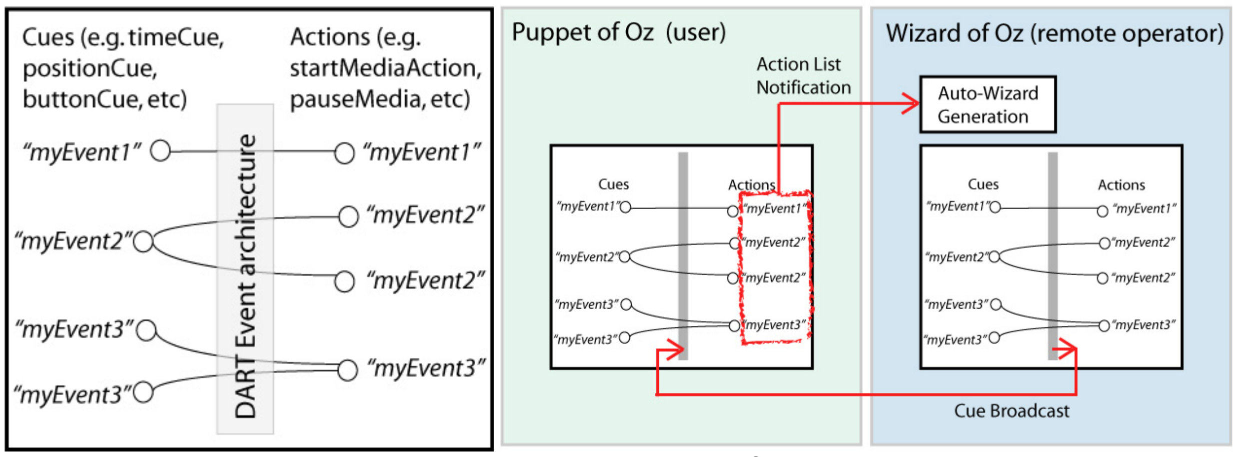
DART Background

For the past few years we have been developing the Designer's Augmented Reality Toolkit (DART), a design environment for prototyping a wide range of applications in augmented reality, mixed reality, and ubiquitous computing. Our goal with DART is to lower the boundary for designers integrating live video, tracking technology, and other sensors within new media applications; one key decision was to integrate this technology into an established and familiar commercial media design environment, Macromedia Director.

DART consists of a Director extension (Xtra) written in C++ to handle the low-level interaction with sensors, trackers, and cameras; and Lingo behaviors (Director's custom scripting language) for manipulating common high-level design features. The DART behaviors are abstractions of system entities that can be configured using simple parameter windows or customized by modifying the Lingo code directly. We seek to support *learnability* through these accessible high-level behaviors that enable designers to quickly get familiar with the environment and its constraints, and to sustain *learning* by permitting designers to peal away layers of abstraction and work directly with code and raw data. At its core DART imbues the philosophy of rapid prototyping and design iteration promoted in this paper; the flexible design environment facilitates an evolution in application logic and content.

Event architecture and networking in DART

DART utilizes a simple event broadcast/subscription model (we use the abstractions of *cues* and *actions*) to communicate between behaviors. A *cue* fires when changes happen within the system: a high-level user interaction (e.g. position cue - "user near site A") or a change in internal state (e.g. time cue - "2 minutes elapsed"). An *action* is executed in response to a particular cue being fired, typically changing media content or the application state. Cues and actions are labeled using unique event names such as "myEvent1"; the designer can set up a cue to respond to a sensor value (e.g. when the GPS device returns a value within some range) and link this to an output action (e.g. play sound file A). An action must subscribe to a cue, using the cue's event name to complete the broadcast/subscription connection. There may be multiple cues that trigger one action or multiple actions subscribed to one cue (Figure 2a). This loose coupling facilitates prototyping; for example, an action initially triggered using a keyboard cue can be replaced by an external device button cue or a more sophisticated cue. More details about the networking and distributed shared memory system are provided in [1].



(a) DART Events **(b) Integration of WOz tools into the event-based system**

Figure 2: (a) A standard event-based architecture in DART: "myEvent1" is a unique name provided for both the cue within system (such as receiving a particular sensor value), as well as the action it generates (such as playing a sound file). (b) DART's WOz tools integrate nicely into the event-based model. Cue Broadcast: Cues fired on the wizard are broadcast locally and to the "puppet". Action List Notification: The puppet notifies the wizard about available actions at run-time to automatically generate a basic button UI on the wizard interface.

Wizard of Oz Prototyping Tools

The Wizard of Oz tools in DART leverage the event broadcast/subscription architecture. To enable WOz communication, the designer adds the "Puppet of Oz" behavior to the machine running the experience. Next, the "Wizard of Oz" behavior is added to the WOz application on a remote machine, along with the IP address of the puppet machine. The two high-level behaviors establish the networking connection and enable the wizard interface to trigger any actions currently available in the user application by simply firing actions locally using the same cue names. The WOz tools employ two forms of communication: cue broadcast (from wizard to puppet) and action list notification (from puppet to wizard) and cue broadcast (from wizard to puppet) [Figure 2b]. By using common naming conventions, cues can be triggered locally or by a remote wizard.

Leveraging the continuous notification of available actions on the puppet, DART can automatically generate a WOz interface consisting of GUI buttons that correspond to the list of possible actions on the puppet. An action subscription list is maintained on the puppet during run-time. Changes to the subscription list are forwarded to the wizard (since events can be added and removed at run-time). The wizard generates a generic button interface (using a simple layout algorithm) and labels each button with the unique event name. As the puppet application runs, the wizard automatically refreshes the corresponding set of buttons. By supporting automatic wizard generation in a high-level behavior (*WizardButtonAuto)*, we aim to lower the barrier for using WOz prototyping as an evaluation strategy.

The wizard interface can also be customized to control the puppet using a mix of built-in Director widgets and DART behaviors. For example, the designer can create a custom button in Director, and then on top of the graphical element place a *WizardButton* script, a parameter of which is a specific action name to launch on the puppet. Alternatively, the designer can place an overhead map image in the application and attach a *MapTracking* behavior that generates synthetic GPS reports that appear to the puppet application to be real GPS reports. Designers can program their own behaviors in Lingo, taking advantage of the networking infrastructure and DART architecture to tailor the wizard interface. All cues sent by the wizard can be received locally by subscribers on the wizard, or only broadcast to the puppet.

One strategy we have used is to integrate part of the user interface of the puppet into the wizard interface so that the wizard operator will experience the same application state as the user. In "The Voices of Oakland" project, described below, we use this strategy to allow the wizard to listen to the same audio segments, follow along with the user, and decide what content to display. Since both interfaces are developed in DART, and because DART uses a model of independent actors communicating via events, any code developed on either the wizard or the puppet interface can be easily ported to the other.

Tools for Visualizing Wizard of Oz Studies

In DART we support the ability to capture data from video cameras, trackers, analogs, buttons, as well as wizard input, for later analysis and playback. The time-synchronized data is stored in Director Casts (collections of media, Lingo scripts, text files, etc. in Director) and can be imported into any DART application. Using integrated playback facilities in DART, the designer can replay the sensor data as it happened during the experiment. The application logic responds to replayed data just as it would to live data, so in playback mode, the designer perceives exactly what the user perceived during the experience.

Experience replay is one visualization strategy; DART also includes tools for visualizing data textually and graphically. *Observers* simply show a text print out of the data based on current time. *DataGraphs* format the replay data into a graphical image. These high-level behaviors require the designer to specify certain parameters, such as the specific replay dataset and the graph boundaries (axis assignment, max values, size of data points, etc). Multiple *DataGraphs* behaviors can be attached to the same image so that multiple streams of data can be visualized on one graph. Visualization tools in DART can show both static, cumulative data as well as a dynamic indication of the data value at any particular time on the abstract clock. For “The Voices of Oakland” experience described below, we visualize the GPS data for each participant’s data on the same image to get an overview of user movement [Figure 5].

Although the numbers collected from WOz experiments can be moved over to Matlab, Excel, or some other common graphing tool, there are certain benefits of integrating these facilities in the DART environment. One advantage is the ability to visualize live data in parallel with previously collected data, enabling real-time analysis of user performance. Embedded visualization takes advantage of DART abstract clock controls. DART replays data on an abstract clock that can be paused, un-paused, set to a particular time, rewound, fast-forwarded, etc. By supporting a number of lightweight visualization tools within the design environment, we enable new sorts of analysis while maintaining consistency with the rest of the design environment.

Case Study: A Mixed Reality Experience in Oakland Cemetery

To illustrate the value of flexible wizard prototyping and analysis tools, we discuss the design of “The Voices of Oakland” project, an audio location-based experience in the historic Oakland Cemetery in Atlanta, GA. Visitors wander through the space and hear an unfolding drama about the history of Atlanta through the voices of its inhabitants [Figure 3]. In our full treatment of “The Voices of Oakland” experience [15], we describe our design considerations including issues of story style (dramatic vs. commentary), story arc (linear vs. non-linear), agency (user vs. system control), medium (visual vs. non-visual media), and technology (location tracking, etc).

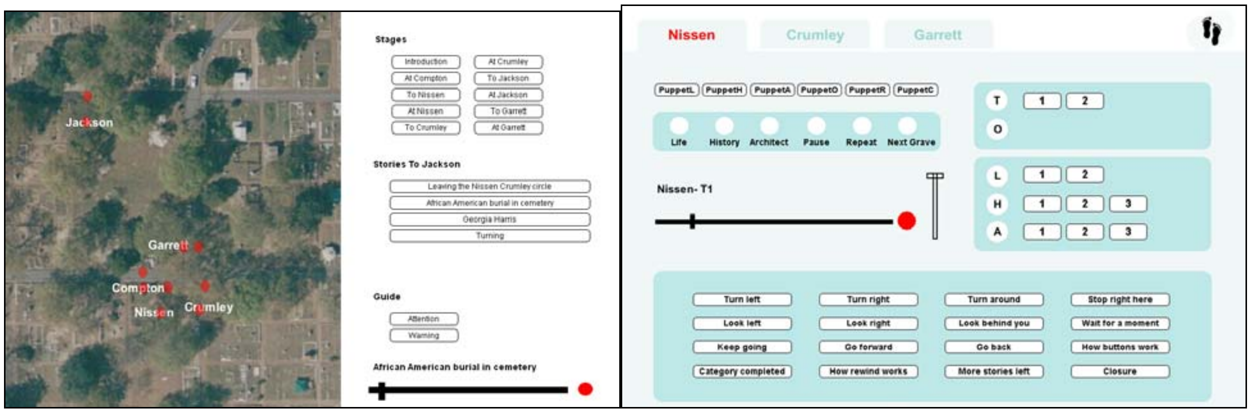
This case study illustrates three roles or levels of responsibility a wizard can have during a design process. In our first iteration, the wizard fully controlled the audio segments the participant heard. In the second iteration, we included buttons for the participant to select content. The button presses were routed to a graphical display in the wizard interface, but the moderator wizard still had to choose the appropriate audio segments based on this feedback. In the third iteration, we handed over the full responsibility to the system for presenting audio based on location and user button interaction; the wizard was present to supervise the experience. The DART WOz tools facilitated an ease of transition between each change in the wizard interface.

Figure 3: Participant experiencing "The Voices of Oakland" in Oakland Cemetery, Atlanta, GA.

First Design Iteration – Wizard as Controller:

In the first generation of "The Voices of Oakland" project, the wizard operator was integral for simulating the experience for several participants. We started with rough audio content and a vague idea of where and when we wanted those stories to be presented as the user moved through the space. To facilitate the *controller* wizard role, we developed two modes of interaction: a map-based interface where the wizard tracks the position of the user and content is triggered when the user moves near a red content zone (Figure 4a-left); and a button-based interface where each button triggers a specific audio segment to play (Figure 4a-right). The map-based interface took several days to create because we had to configure the map image to correspond with the correct GPS coordinates and we had to link the audio content to a particular location in the cemetery. The button-based interface was very quickly generated using DART's *WizardButtonAuto* behavior.

In our pilot study, we were interested in evaluating the wizard interface as well as the participant experience. Our evaluation consisted of two different wizards (both were script writers for the experience) and two users. Aside from demonstrating how the wizard interface worked, we provided no specific instructions for the wizard operator how to create the experience for the visitor; the operators could choose to either simulate the GPS tracking or push buttons to directly trigger the media content. During the study we found that our wizard operators preferred the buttons, as opposed to the map, ostensibly because it put them closer to the content and enabled them to create a more compelling experience for the visitor. (However, because the location-based implementation was fairly crude at this point, this observation should be taken with a grain of salt.) The tour participants provided feedback about the type of content and length of audio segments, and we incorporated many ideas into the next iteration.



(a) (b)

Figure 4: (a) Wizard interface used for the first design iteration. As a "controller", the wizard has the choice to simulate user position on the map (left) or to trigger audio content directly with auto-generated buttons (right). (b) Wizard interface used in the second and third design iteration. The wizard observes the user's button interaction in the six circles in the upper left and activates audio content with the buttons (story segments on the right and navigational segments at the bottom).

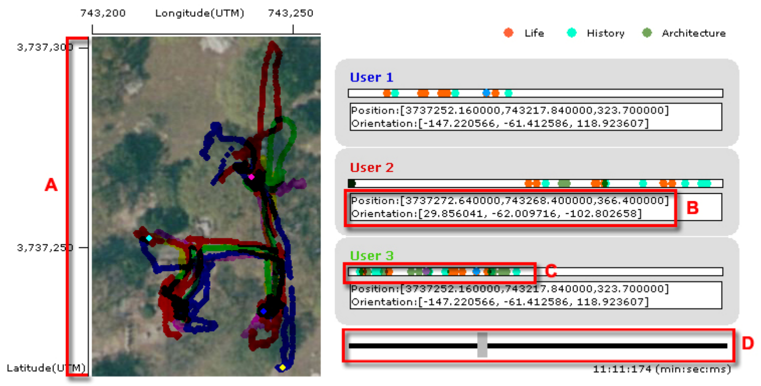
Second Design Iteration – Wizard as Moderator:
After the initial pilot testing, several changes were made to the experience and to the role of the wizard. The script was divided into short audio dialogues and organized into categories (Life Stories, History, and Architecture). To simulate user control, we provided the user with a hand-held controller with physical buttons to correspond with the content categories. The visitor could listen to short audio stories within a category and then dig deeper, or choose a different category while at any particular grave site.

The new wizard interface [Figure 4b] incorporated Director's built-in GUI widgets and included a set of buttons (near the bottom of 4b) for navigating the visitor if the primary instructions were not adequate. We chose to route the user's button interaction through the wizard interface, and situate the wizard as a *moderator*. We had questions about the users' expectations about the interaction, and we wanted to control for interface issues such as multiple button pressing, while providing an engaging experience. The moderator would see the button interaction in the wizard interface (and the user's position in the physical space) and determine the appropriate content. Although it took longer to create this custom wizard interface, the DART WOz tools enabled us to easily co-evolve the wizard to match the new user interaction and to support the moderator wizard role.

During an outdoor Fall Festival in the cemetery, we informally tested the experience with about fifteen different members of the public. For the most part, our four different wizard operators simply relayed user button presses, providing an indication that the user button interaction was sufficient for controlling the audio content at a particular grave. We did not record any raw data (GPS) or try to visualize the results, in part, because we were not doing a formal evaluation, although we realized it would have been useful.

Third Design Iteration – Wizard as Supervisor:
For the third version of "The Voices of Oakland", we allowed participant interaction with the button device to directly impact changes in content. The visual wizard interface remained the same (Figure 4b), but the wizard's role shifted to that of a *supervisor*. The wizard could observe the user's button presses and override the user's content choices, if necessary. The wizard still communicated to the system when a participant was in range of a particular grave and controlled the ancillary navigational content, if necessary.

We used DART's capture facilities to record all the sensor data (GPS, head-rotation, button presses) and wizard actions during a formal user evaluation. Even though we were not using the GPS and head-rotation data in the application, we collected the data to help us close the gap between the wizard-simulated experience and a working application. Using the replay and visualization tools in DART, we created interactive graphs to help analyze the WOz study (Figure 5).



```
Position:[3737252.160000,743217.840000,323.700000]
Orientation:[-147.220566, -61.412586, 118.923607]

Position:[3737272.640000,743268.400000,366.400000]
Orientation:[29.856041, -62.009716, -102.802658]

Position:[3737252.160000,743217.840000,323.700000]
Orientation:[-147.220566, -61.412586, 118.923607]

11:11:174 (min:sec:ms)
```

Figure 5: Visualization of "The Voices of Oakland" experience created with a handful of DART behaviors (*DataGraphs, Observers, TimeSlider*) inside Director. a) GPS data for 5 participants with dynamic circles showing the user's position at a particular time. b) Textual representations of GPS location and head rotation data. c) Graph of button interaction over time. d) Slider for control of DART's abstract time.

The visualization tools in DART are an initial attempt at supporting WOz study analysis. In this example, the GPS data from multiple participants in the experience is displayed above a satellite image of the cemetery. For each participant we could print the head-rotation values and graph the button interaction over time. We included a time slider in our visualization enabling us to scrub the time value and see dynamic updates in the graphs. Our visualization helped us understand how GPS and head rotation data might be useful in our system design (while it might have been useful to provide a graphical view of the rotation values, we opted for a simpler textual representation until the graphical view is clearly needed). We refer readers to the full paper on the Oakland experience for more details about the study results [15]. During future WOz evaluations we also plan to explore visualizing prior participant data during the study of live participants for real-time predictive feedback on user behavior.

Conclusions

This paper emphasizes the importance of explicit support for the Wizard of Oz simulation method with pervasive computing prototyping environments. By providing easy-to-use high-level tools for wizard interface creation and data visualization, we hope to lower the barrier for designers and researchers to explore a variety of potential uses of WOz simulation throughout the design cycle.

We learned many lessons during our experimentation with WOz simulation in the Oakland project. Clearly, the nature of the application will dictate what facilities are needed. The ability to easily try different WOz approaches was very useful, given the design space we were exploring. If we were doing visual AR and/or spatialized audio where accurate location was integral, the map based wizard interface may have been more useful, but we may not have had as many other useful options for wizard controls.

The automatic wizard interface generation appears to be most useful for early design exploration (e.g., we used the automatically generated interface during our early informal studies, but did not use them during our formal WOz studies, because customized WOz features were more useful). The embedded visualization tools are well suited for intermediate stages of design and novel wizard roles, where much can be learned about the user's preferred interaction and the limitations of the technology. In this paper, we have highlighted a relatively uncommon *moderator* role for the wizard, to help transition WOz prototypes into final applications, and to collect realistic evaluation data for evaluating system components and sensors. The rapid prototyping philosophy of DART and our WOz tools encourages designers to explore a variety of roles with different degrees of responsibility through a design process.

There are many significant considerations, raised by many researchers over the past decade, which designers must be conscious of when designing WOz experiments. In particular, the wizard interface should be designed to consider the wizard operators' perceptual, cognitive, and motor skills, just as with any user interface. By reducing the demand on the wizard, they are able to pay more attention to the user and the environment, perform the wizard activities more easily, and contribute their observations to the evaluation of the experience. For the WOz method to be effective, it should be feasible to bridge the gap between the wizard's role and actual system implementation. We believe that the integration of WOz prototyping tools into the programming environment is vital to support this transition.

References

1. MacIntyre, B., Gandy, M., Dow, S., Bolter, J.D. "DART: A Toolkit for Rapid Design Exploration of Augmented Reality Experiences", In ACM Symposium on User Interface Software and Technology (UIST'04), 2004, pp 197-206.

2. Dow, S., Lee, J., Oezbek, C., MacIntyre, B., Bolter, J.D., and Gandy, M. "Wizard of Oz Interfaces for Mixed Reality Applications," In Extended Abstracts of SIGCHI Conference of Human Factors in Computing Systems (CHI'05), 2005, pp 1339-43.

3. Hudson, S., Fogarty, J., Atkeson, C., Avrahami, D., Forlizzi, J., Kiesler, S., Lee, J. and Yang, J. "Predicting Human Interruptibility with Sensors: A Wizard of Oz Feasibility Study," In SIGCHI Conference on Human Factors in Computing Systems (CHI 2003), 2005, pp 257-264.

4. Voida, S., Podlaseck, M., Kjeldsen, R., and Pinhanez, C. "A Study on the Manipulation of 2D Objects in a Projector/Camera-Based Augmented Reality Environment," In SIGCHI Conference of Human Factors in Computing Systems (CHI'05), 2005, pp 611-620.

5. Li, Y., Hong, J., and Landay J. "Topiary: A Tool for Prototyping Location-Enhanced Applications," In ACM Symposium on User Interface Software and Technology (UIST'04), 2004, pp 217-226.

6. Lyons, K. "Improving Support of Conversations by Enhancing Mobile Computer Input," Ph.D. Thesis, 2005.

7. Tran, Q., Mynatt, E. "What Was I Cooking? Towards Deja Vu Displays of Everyday Memory," Georgia Institute of Technology Technical Report GIT-GVU-TR-03-33.

8. MacIntyre, B., Bolter, J.D., Moreno, E., and Hannigan, B. "Augmented Reality as a New Media Experience," In International Symposium on Augmented Reality (ISAR'01), 2001, pp 197-206.

9. Koleva, B., Adams, M., Taylor, I., Benford, S., Fraser, M., Greenhalgh, C., Schnädelbach, H., vom Lehn, D., Heath C., Row-Farr, J. (Nottingham and Blast Theory) "Orchestrating a Mixed Reality Performance," In SIGCHI Conference of Human Factors in Computing Systems (CHI'01), 2001, pp 38-45.

10. Dahlback, N., Jonsson, A., and Lars, A. "Wizard of Oz Studies: Why and How," In International Workshop on Intelligent User Interfaces (IUI'93), 1993, pp 193-200.

11. Molin, L. "Wizard of Oz Prototyping for Cooperative Interaction Design of Graphical User Interfaces," In SIGCHI Nordic Conference on Human-Computer Interaction (NordiCHI'04), 2004, pp 425-428.

12. Maulsby, D., Greenberg, S., and Mander, R. "Prototyping an Intelligent Agent through Wizard of Oz," In ACM SIGCHI Conference on Human Factors in Computing Systems (CHI'93), 1993, pp 277-284.

13. Klemmer, S., Sinha A., Chen J., Landay J., Aboobaker N., Wang A. "SUEDE: A Wizard of Oz Prototyping Tool for Speech User Interfaces," In ACM Symposium on User Interface Software and Technology (UIST'00), 2000, pp 1-10.

14. Balbo, S., Coutaz, J., and Salber, D. "Towards Automatic Evaluation of Multimodal User Interfaces," In International Workshop on Intelligent User Interfaces (IUI'93), 1993, pp 201-208.

15. Dow, S., Lee, J., Oezbek, C., MacIntyre, B., Bolter, J.D., and Gandy, M. "Exploring Spatial Narratives and Mixed Reality Experiences in Oakland Cemetery," In ACM SIGCHI Conference on Advances in Computer Entertainment (ACE'05), 2005, pp 51-60.

Author Bios:

Steven Dow is a PhD student in Human-Centered Computing in the College of Computing at the Georgia Institute of Technology. His research interests include design tools, HCI, ubiquitous computing, mixed reality, and experience design. Contact her at the College of Computing, GVU Center, Georgia Tech, Atlanta, GA 30332-0280; steven@cc.gatech.edu.

Blair MacIntyre is an assistant professor in the Georgia Institute of Technology's College of Computing and the GVU Center. His research interests include understanding how to create highly interactive augmented-reality environments, especially those that use personal displays to augment a user's perception of his or her environment. He received his PhD in computer science from Columbia University. Contact him at the College of Computing, GVU Center, Georgia Tech, Atlanta, GA 30332-0280; blair@cc.gatech.edu.

Jaemin Lee earned her masters degree in Human Computer Interaction in the College of Computing at the Georgia Institute of Technology. Her research interests include HCI, ubiquitous computing, and context-aware computing. Contact her at the College of Computing, GVU Center, Georgia Tech, Atlanta, GA 30332-0280; jaemn@cc.gatech.edu.

Christopher Oezbek is a PhD student in Software Engineering at the Free University Berlin after having completed his MS in Computing at Georgia Tech in 2004. His research interests include documentation processes, API usability and discovery, and the Open Source development process. Contact him at the Working Group Software Engineering, Takustr. 9, D-14195 Berlin, Germany; oezbek@inf.fu-berlin.de.

Jay David Bolter is the Wesley Chair of New Media at the Georgia Institute of Technology. He is the author of several books on hypertext, media theory, and digital design. He has a Ph.D. in Classics (University of Toronto). With the AEL collaborators at Georgia Tech, he is helping to build Augmented Reality (AR) systems to stage dramatic experiences for entertainment and education. Contact him at the School of Literature, Communication, and Culture, Georgia Tech, Atlanta, GA 30332-0165; jay.bolter@lcc.gatech.edu.

Maribeth Gandy is a Research Scientist with the Interactive Media Technology Center at the Georgia Institute of Technology where she has worked since completing her MS in Computer Science in 2000. She is currently pursuing her PhD with a focus on Augmented Reality and HCI. Her other research interests include mobile/wearable and ubiquitous computing, universal design, and computer audio. Contact her at the Interactive Media Technology Center, Georgia Tech, Atlanta, GA 30332-0280; maribeth@imtc.gatech.edu.