Towards a Unified Interaction Framework for Ubicomp User Interfaces

Jason I. Hong, Scott Lederer, Mark W. Newman
Group for User Interface Research
University of California, Berkeley
Berkeley, CA 94720
{jasonh, lederer, newman}@eecs.berkeley.edu

ABSTRACT
The remarkable success of the personal computing era is largely attributable to the WIMP desktop interaction framework. We identify a set of core design techniques embodied by the WIMP desktop, specifically aggregators, objects, commands, and selectors, and discuss their applicability to the design space of ubiquitous computing user interfaces. We offer some potential research directions for exploring this approach and present a set of open research questions.

INTRODUCTION
The invention of the GUI enabled more people to use computers in more situations and tasks than was possible with command-line interfaces. However, the modern GUI did not appear fully-formed like Athena from the head of Zeus. Watching videos of pioneering graphical systems such as Sutherland's Sketchpad [7] and Engelbart's NLS [3] is insightful not only for seeing what these systems enabled, but also what they lacked. For example, these systems did not have widgets such as buttons and scrollbars, overlapping windows, icons, menus, as we know them today. A person using these systems today would likely feel that the interaction was stilted in the same way modern speakers of English feel when they read old English.

Here we differentiate between GUIs, WIMP, and the desktop metaphor. By GUI, we mean all graphical user interfaces broadly construed. WIMP is the familiar subset of GUIs comprised of Windows, Icons, Menus, and Pointers. The desktop metaphor is the conceptual model that ties the elements of WIMP together. It should be noted that the WIMP desktop was not created arbitrarily, but resulted from the careful application of refined usability design principles to the GUI design space [5]. By offering a consistent interaction framework, the WIMP desktop empowered users to intuitively transfer basic interaction skills across a range of applications and devices.

For ubiquitous computing to surpass the success of the personal computing era, we believe that Ubicomp User Interfaces (UUIs) will require an analogous unified interaction framework (see Figure 1). Although we as a community do not currently know what the full design space of UUIs will be, we are beginning to discern its dimensions as UUI components emerge. There is a tendency in research to invent novel principles and guidelines to shape the future of ubicomp interaction. While ubicomp certainly presents many new constraints and opportunities, it would be shortsighted to abandon the principles behind the WIMP desktop, a successful design solution. The questions to ask then are, what interaction problems did the WIMP desktop successfully address; how did it address them; which of these problems apply to the design space of UUIs; and can we apply similar solutions for UUIs?

The rest of this position paper is divided along these questions. First, we look at some of the problems that the WIMP desktop solved and the design principles used in addressing these problems. Second, we examine the pertinence of these problems to the design space of UUIs. Lastly, we present research directions for exploring the applicability of WIMP's design solutions to these problems in the UUI design space.

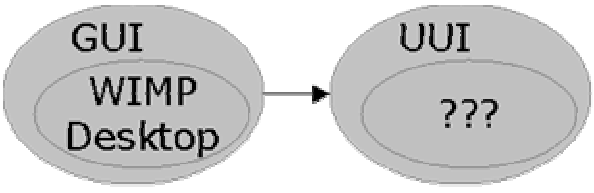

Figure 1 – The WIMP Desktop is the familiar subset of GUIs. Can we design an analogous interaction framework for Ubiquitous User Interfaces (UUIs)?

DECONSTRUCTING THE WIMP DESKTOP

Decomposing the WIMP desktop into its four constituent parts[1], we see that:

- *Windows* are a unit of aggregation for commands and data, providing encapsulation for and multiplexing of tasks;
- *Icons* represent objects (nouns) that can be operated on (ex. files) or executed (ex. programs). Icons can provide feedback on their current state and communicate some notion of what you can do with them (ex. trashcans);
- *Menus* provide a standard way of executing commands (verbs). Menus, along with other approaches such as buttons and toolbars, make the set of valid commands visible to end-users;
- *Pointers* provide a selection mechanism, allowing end-users to choose a set of objects and execute a series of commands on them.

Evaluating these aggregators, objects, commands, and selectors (AOCS) according to Norman and Draper's design principles of visibility, conceptual models, good mappings, and feedback [6], we see that:

- Windows provide a way of visualizing and managing active tasks;
- Icons (and widgets in general) provide visibility for what targets are available, what targets are selected, feedback on the current system state (sometimes as direct manipulation feedback), and a focus for keyboard and mouse input;
- Menus (and widgets in general) provide a visible state of what commands are possible, visual affordances for how to execute those commands; and a reliable way of executing commands (ex. cannot misspell a menu command);
- Pointers provide some visible feedback about what operations are possible (ex. resize window or the I-bar cursor for inserting text);
- The consistent application of these mechanisms across systems has engendered a sufficiently accurate conceptual model in the user population.

AN APPROXIMATE DESIGN SPACE OF UBICOMP USER INTERFACES

One important unanswered question is, what is the full design space of UUIs from which a smaller subset can be drawn? Although there are no definitive answers yet, current research and commercial trends suggest that UUIs are likely to include:

- A richer range of input types, including more natural modes of communication such as speech and sketching [1], and implicit input through sensors [2];
- A wider range of output types, including multiple small displays both portable and embedded, aural feedback, haptic feedback, and ambient feedback through channels such as sound and peripheral vision;
- Multimodal input and output across multiple portable and embedded devices;
- Interaction scaled over space, time, devices, and users, where ongoing tasks will be in various states of activity across time and space, and will involve multiple devices and users [1];
- Physical spaces and objects entwined with virtual ones.

Given this design space, how might we apply AOCS to satisfy the established design principles of visibility, feedback, good mappings, and robust conceptual models? We present some possible directions as seeds for potential solutions.

Regarding aggregation, there are a number of object classes—such as documents, applications, devices, users, data, and physical spaces— that might be aggregated to help users cope with the scale of information and interaction intrinsic to ubicomp. Shifting the nexus of interaction from keyboard, mouse, and display to the world at large, though, suggests that we may need aggregations of a more conceptual nature than the visual bounding box of windows, such as *spaces*, *tasks*, *groups* of people, and contextual *situations.* If such abstract aggregations are feasible, then providing consistent feedback, visibility, and mappings across the

[1] The modern WIMP desktop incorporates many mechanisms and design principles, but in the interests of space we have decomposed only those behind the acronym itself.

UUI design space to support interaction with and awareness of them will be a considerable design challenge.

Similarly, a consistent notion of objects, commands, and selectors for UUIs is not readily apparent. Objects might be virtual or physical, and objects of both natures might be semantically or causally entwined, such as manipulating a virtual object via a physical artifact, or including a reference to a physical object or space in an aggregation of task-oriented virtual objects. Either way, the objects and commands available in a given UUI will vary between spaces and situations, and may often be unobvious without adequate affordances and feedback. While routine behavior may enable a user to understand which physical and virtual objects are operable under which commands in a given situation, a unified framework must also empower users unfamiliar with those spaces and situations.

A possible solution may be sitting just in front of our noses: the GUI. The continuing proliferation of GUIs across all manner of form factors might serve as a universal platform for communicating feedback and affordances through visual representations of objects and commands. While aural, tangible, and ambient interfaces promise to enrich the ubicomp user experience, their deployment does not necessitate forsaking the considerable design experience and user familiarity intrinsic to the modern GUI. Nonetheless, careful design of coherent visual feedback and affordances across spaces, devices, and tasks will be necessary to avert visual overload.

Even if a user knows which objects and commands are operable, the system needs a way of disambiguating multimodal commands meant for one object but applicable to many [2]. For example, should the spoken imperative, “Warmer,” apply to the room you are in or the teacup you are holding? There is little indication that systems emerging over the next few years will incorporate enough intelligence and finesse to reliably disambiguate command targets under general conditions. The traditional solution to the selection problem is a manual selection mechanism, whereby the user selects a target, receives feedback that the correct target is selected, and then executes the command thereon (noun-verb). The tool-based approach, where the command is selected before the target, has also worked under the right circumstances (verb-noun). In either case, the user’s responsibility for specifying the correct target is mitigated by feedback about which targets are currently selected. A unified UUI framework might employ manual selection techniques by establishing universal haptic, aural, and visual feedback on portable and embedded GUIs that indicate which physical and virtual objects are currently selected.

OPEN QUESTIONS FOR A UNIFIED UBICOMP INTERACTION FRAMEWORK

We have presented some research directions for applying AOCS to the UUI design space. However, a host of open research questions remain, as these approaches barely scratch the surface of the problem.

The first set of questions we present look at the utility of applying AOCS to UUIs. For example, there have been many changes between GUI and UUI in terms of tasks, technology, and social needs. With respect to tasks, the WIMP desktop was designed for office work, whereas many people expect ubicomp to encompass all aspects of life from cradle to grave. With respect to technology, the WIMP desktop was designed for keyboard, mouse, and display, all connected to a single computer, whereas ubicomp is likely to make use of a rich range of physically distributed inputs and outputs. With respect to social needs, there are many questions about the relationship between ubicomp and individual privacy and accessibility. It is not clear if AOCS, designed for such different constraints, is really applicable here.

If AOCS *is* applicable, is it a useful style of interaction for ubicomp? Or are other interaction styles, such as natural language and dialog-based, more appropriate? For example, AOCS deals more with explicit interaction rather than implicit interaction, as envisioned by many context-aware systems. Could such a style of interaction encompass both implicit and explicit interaction?

The most important question to ask here is, what kinds of problems would AOCS solve for ubicomp? What kinds of applications would it enable, and for whom? Likewise, what kinds of applications would it make it hard to build?

The second set of questions we pose look at the barriers to implementing AOCS for UUIs. Assuming that AOCS are useful for UUIs, how do we design the framework to embody principles such as visibility, robust conceptual models, good mappings, and feedback? Similarly, is a unifying metaphor like the desktop metaphor required?

Furthermore, a unified interaction framework needs to support developers, not just users. Developers need an open framework on top of which they can design, prototype, evaluate, and implement reliable applications. A robust event model for distributed applications is needed [4], as are toolkits that incorporate good design principles and make it easier to do "the right thing."

Another question to ask is, what mistakes did the WIMP make, and can we avoid repeating those mistakes? For example, the WIMP desktop does not preclude groupware; however, all of the implementations out there make it quite difficult.

Lastly, given all of these requirements, constraints, and open questions, what are some simple first steps the research community can take towards a unified UUI framework? Is there a niche that we can start in and expand from, similar to how the WIMP desktop started in office environments? Is there a simple sharable testbed with which we can experiment with various interaction styles and compare results?

In summary, we have suggested taking the idea of aggregations, objects, commands, and selectors from the successful WIMP desktop, and looked at how these ideas might be applied to Ubicomp User Interfaces, and what some of the issues are in making this happen.

References

1. Abowd, G.D. and E.D. Mynatt, Charting Past, Present, and Future Research in Ubiquitous Computing. *ACM Transactions on Computer-Human Interaction, Special Issue on HCI in the New Millennium* 2000. **7**(1): p. 29-58.
2. Dey, A.K., D. Salber, and G.D. Abowd, A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human-Computer Interaction (HCI) Journal* 2001. **16**(2-3).
3. Engelbart, D.C. and W.K. English. A Research Centre for Augmenting Human Intellect. In Proceedings of *Fall Joint Computing Conference*. Thompson Washington DC. pp. 395-410 1968.
4. Johanson, B., A. Fox, P. Hanrahan, and T. Winograd, The Event Heap: An Enabling Infrastructure for Interactive Workspaces. 2000. http://graphics.stanford.edu/papers/eheap/index.html
5. Johnson, J., T.L. Roberts, W. Verplank, D.C. Smith, C.H. Irby, M. Beard, and K. Mackey, The Xerox Star: A Retrospective, *IEEE Computer*, vol. 22(9): pp. 11-29, 1989.
6. Norman, D. and S.W. Draper, *User Centered System Design*. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc., 1986.
7. Sutherland, I., *Sketchpad - A Man-machine Graphical Communication System*, Unpublished PhD Dissertation, Massachusetts Institute of Technology, 1963.