Understanding and Designing Surface Computing with ZOIL and Squidy

Hans-Christian Jetter, Werner A. König, and Harald Reiterer
HCI-Group, University of Konstanz
Box-D73, Universität Konstanz, 78457 Konstanz
{jetter, koenigw, reiterer}@inf.uni-konstanz.de

ABSTRACT
In this paper we provide a threefold contribution to the Surface Computing (SC) community. Firstly, we will discuss frameworks such as "Reality-based Interaction" which provide a deeper theoretical understanding of SC. Secondly, we will introduce our ZOIL user interface paradigm for SC on mobile, tabletop or wall-sized devices. Thirdly, we will describe our two software tools "Squidy" and "ZOIL UI Framework" which have supported and facilitated our iterative design of SC prototypes.

ACM Classification Keywords
H.5.2 [Information Interfaces and Presentation]: User Interfaces – Input devices and strategies, Graphical user interfaces (GUI), Interaction styles.

INTRODUCTION
Products such as Microsoft Surface and the Apple iPhone have moved Surface Computing (SC) and Multitouch Interaction into the focus of consumers', researchers' and designers' attention. The ongoing media coverage reveals the high expectations towards SC, which go beyond the obvious change from mouse and keyboard to interacting by touch, finger gestures or physical artifacts. In fact SC is believed to lead into a more "natural" or "intuitive" future of human-computer interaction imparting a yet unknown feeling of directness, learnability, user empowerment and efficiency.

As HCI researchers, we believe that these high expectations can only be met when three preconditions are satisfied: Firstly, that the true nature and full potential of SC is properly described and understood in theoretical frame-works, which acknowledge the fundamental change from desktop computing to "embodied interaction" (EI) [3] or "reality-based interaction" (RBI) [9] that we are witnessing today. Secondly, that taking SC seriously means to question well-established fundamentals of today's personal computing, e.g. the application-centered desktop metaphor, the hierarchical file and folder hierarchy, and the list- and hyperlink-dominated World Wide Web [13]. Thirdly, that powerful hardware and software prototyping tools for SC are essential for an extended iterative research and design cycle [6].

In the following we will illustrate how we have addressed these issues in our ongoing research projects and how our "ZOIL" and "Squidy" software tools contribute to SC.

Workshop - Multitouch and Surface Computing
CHI 2009, April 4 – 9, 2009, Boston, MA, USA

UNDERSTANDING SURFACE COMPUTING
Discussing and understanding SC can be taken out at very different levels of abstraction: On a motor behavior and control level, SC has the benefit of a unified motor and display space located on the touch surface [1] which allows for a bimanual direct and absolute pointing on multiple user interface objects.

While direct pointing certainly is the most fundamental characteristic of SC, "directness" in the sense of Hutchins et al.'s "Direct Manipulation" [8] cannot be provided solely on the input/output layer. For this purpose higher level principles of interface architecture and interaction design have to be introduced.

Hutchins et al. have described Direct Manipulation in terms of two competing interface metaphors: the indirect "conversational metaphor" and the direct "model-world metaphor". Conversational interfaces are "a language medium in which user and system have a conversation about an assumed, but not explicitly represented world". In a model-world interface, "the interface is itself a world where the user can act, and which changes state in response to user actions. The world of interest is explicitly represented and there is no intermediary between user and world". We believe that such Direct Manipulation in a model-world interface is an optimal solution for SC design. However, it remains difficult or impossible to achieve using contemporary UI architectures and interaction models, which are often focused on rigidly structured conversations with Hypertext or dialog-driven "single point of control" interaction ([3], p.51).

The notion of computing as Direct Manipulation within a model-world metaphor has been further developed in the

context of mobile & ubiquitous computing. Theoretical frameworks such as EI or RBI aim at understanding and innovating this new generation of user interfaces that draw from the previously neglected human skills for acting in the physical and social world. EI and RBI expand the "model-world" from inside the computer into the real world, creating a physical space of interaction or ambient intelligence surrounding the user which is augmented with computational power and information using intelligent artifacts such as phidgets, mobile devices or tabletops. In this way users can combine both their virtual computer-based powers and their bodily and social skills from the real world. Therefore, unlike the overly concrete real-world metaphor of the "desktop" in today's GUIs [14], this new generation of interfaces draws from more abstract and essential "themes of reality such as users' understanding of naïve physics, their own bodies, the surrounding environment, and other people" [9].

Such an "embodied" perspective on SC reveals and tries to make use of the post-cartesian closeness between human cognition and action, e.g. in the fields of visual perception, eye, head & body movement, gestures or spatial memory. The degree to which SC will succeed to create such an authentic overlap between the real and virtual world will define how "natural", "intuitive" or "logical" computing will appear to us in the future.

We would like to emphasize the importance of such a holistic view of SC encompassing not only SC's characteristics on the input/output layer and its need for Direct Manipulation, but also its emerging role as a critical component of an embodied and reality-based future. Only if SC is viewed through the lens of EI and RBI its full potential for a more natural HCI will become obvious.

THE ZOIL USER INTERFACE PARADIGM

The ZOIL user interface paradigm is an ongoing research effort to transform our theories and experiences in the fields of interaction design, visual information seeking, information visualization and zoomable user interfaces into an application- and platform-independent user interface concept. Backed up by concise design principles and supporting software frameworks, ZOIL (=**Z**oomable **O**bject-Oriented **I**nformation **L**andscape) is introducing a new post-WIMP visual user interface paradigm for SC on mobile, tabletop or wall-sized devices. ZOIL user interfaces integrate all types of information items with their connected functionality and with their mutual relations in an object-oriented visual workspace following a simple and consistent interaction style based on zooming, panning and Direct Manipulation. This visual workspace named the "information landscape" serves as an integrated work environment, which replaces the browser, the file system and the application windows of today's WIMP GUIs (see figures on last page). An overview about the fundamental ZOIL concepts and design principles has been provided in [13] and is based on a broad range of previous work (e.g. [12], [15], [5], [10]). ZOIL is subject to ongoing changes and undergoes cycles of design, evaluation and reformulation based on building and testing prototypes such as [11]. A video showing how ZOIL can be applied within the domain of Personal Information Management on high-definition television, multitouch tabletops and display walls is available online at [13].

In the following we will briefly focus on two selected features of ZOIL which are especially relevant with regard to the previously introduced theoretical frameworks.

Object-Oriented User Interface (OOUI) – In OOUIs "object-orientation" (OO) is not a question of the employed language and architecture for implementation, but of the way how content and functionality is visualized on the user interface and can be accessed and manipulated by the user. Collins describes the features of OOUIs in [2]: "Users perceive and act on objects. Users can classify objects based on how they behave. In the context of what users are trying to accomplish, all the interface objects fit together into a coherent overall representation".

For Surface Computing an important characteristic of OOUIs is the way how they succeed in providing complex functionality as a "collection of cooperating objects and views of objects" [17], which are suitable for (simultaneous) Direct Manipulation with touch and gestures. In OOUIs even complex tasks are carried out by creating, dragging, resizing or connecting objects in a model-world, therefore empowering the user to perform in their own way or to innovate without having to follow a rigid conversational or sequential application structure. If properly applied during UI design, OO modeling techniques such as inheritance and polymorphism achieve a maximum degree of consistency and interoperability in the behavior of the user interface objects allowing for an easy integration of new views or new kinds of objects without fundamental changes.

Zoomable Information Landscape – In a ZOIL user interface all objects appear as visual representations in the information landscape. This landscape is not limited to the visible screen size but resembles a virtual canvas of infinite size and resolution. All objects, their functionality and content can be accessed by panning to the right spot in the information landscape and zooming in.

As suggested by Perlin & Fox [18], ZOIL uses "semantic zooming" meaning that the geometric growth in display space is not only used to render more details, but to reveal more content and functionality. For example zooming into objects like documents might turn them into readable and editable areas, while zooming into videos might reveal more metadata and player controls.

For SC such Zoomable User Interfaces (ZUIs) are of great use, since they allow the natural presentation of great amounts of information without having to introduce page navigation, popup- or application-windows or other means

of disruptive and occluding "details-on-demand" techniques. The cognitive effort for acting and navigating in the information space is further minimized by "natural" gestures for zooming and panning and by exploiting the users' visual-spatial orientation and manual skills instead of making them recall invisible navigation histories or page hierarchies.

TOOL SUPPORT WITH SQUIDY AND ZOIL

After having introduced the ZOIL user interface paradigm, we will illustrate how a prototype of a SC system following ZOIL can be built using our Squidy interaction library and our ZOIL UI framework. Fig. 1 shows the system architecture based on the layer model introduced in ([3], p.141).

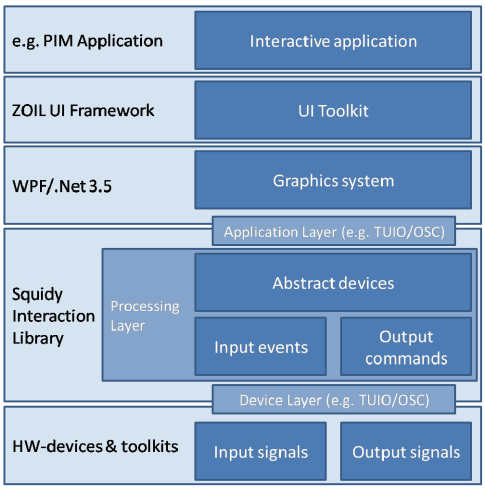

Figure 1: System Architecture

SQUIDY INTERACTION LIBRARY

In the absence of a standard multitouch driver, SDK or protocol, interaction designers in the context of SC are confronted with various monolithic systems, although the basic functions of the systems are the same. For instance designing a specific touch gesture in a cross-platform application, requires the integration of different interfaces e.g. the Apple iPhone SDK, the NUIGroup Touchlib, and the Microsoft Surface SDK. Although they share the same general tasks – identification of contact positions, data preprocessing and transmission – they differ in their API, their programming language and architecture. Thus, designers additionally need to learn and integrate diverse interfaces for every application. They should ideally be experts for all systems and layers – from physical devices over drivers and toolkits up to the applications. Integrating further non-standard input devices or tracking techniques such as tangible user interfaces, pen input or freehand tracking systems which also come with their own tracking library is almost unmanageable.

Our objective is to reduce the additional workload and complexity for the interaction designer and researcher caused by the heterogeneous landscape of device drivers and toolkits. Therefore, we introduce the interaction library "Squidy" which unifies diverse device toolkits in a common library and provides a central user interface for visual data flow management as well as device and data filter configuration. Squidy has a three-layer architecture separating the specific drivers and toolkits in the *device layer* (lower layer) from the data filtering, processing and interpretation in the *processing layer* (middle layer). The ZOIL user interface framework and instances of individual applications are linked to the *application layer* (upper layer).

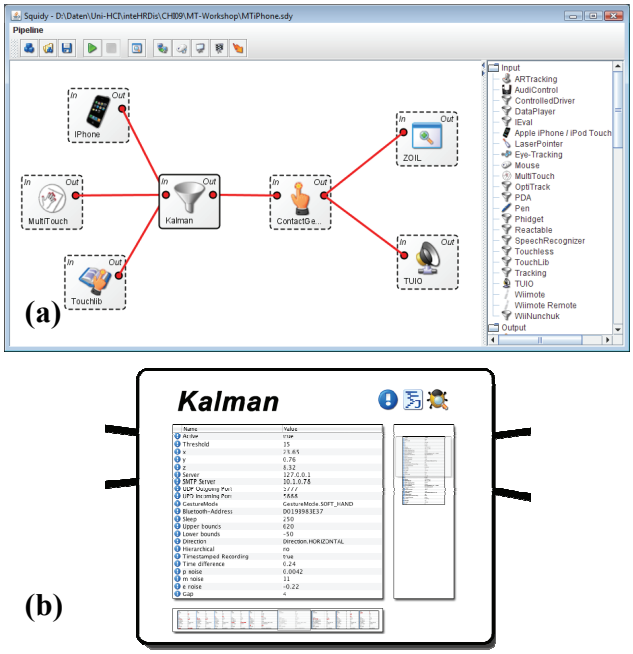

Figure 2: (a) Squidy user interface: visual data flow management and filter configuration by semantic zooming, (b) zoomed Kalman filter node with related parameters.

For the communication between the layers we generalized the various kinds of input and output data to a hierarchy of well-defined atomic data types based on the primitive virtual devices introduced by Wallace [19]. Replacing devices (e.g. switch from Surface to iPhone) does not necessarily affect the applied interaction techniques (e.g. "selection by dwelling") defined in the processing layer or the concrete application linked to the upper layer. The independent layer approach in combination with the generalization of data types enables the fusion of heterogeneous devices and toolkits in the common interaction library Squidy aiming at reducing the development complexity and effort. Squidy also empowers the designer to realize multimodal interaction by visually combining different input and output modalities in the Squidy user interface without the need of programmatical changes. Based on a client-server architecture, Squidy provides different network (TCP/IP, UDP, TUIO/OSC) and programming interfaces (Java, C++, .Net 3.5 API) for the upper and lower layer covering a wide range of input and

output devices and standard WIMP-applications as well as individual multi-point applications. Currently, it supports the Apple iPhone SDK and the NUIGroup Touchlib for multitouch interaction, the ART DTrack and the NaturalPoint OptiTrack for finger gestures [4] and body-tracking, the libGaze for mobile eye-tracking [7], the Microsoft Touchless SDK for mid-air object tracking, the Phidgets API for physical prototyping and self-developed components for laser-pointer [16], multitouch, Nintendo Wii Remote and TUI interaction.

While the upper and lower layers provide interfaces and organize data transmission, the interaction logic is defined at the processing layer which is visualized by a directed graph in the Squidy user interface (see Figure 2a). The nodes represent input and output devices as well as data filters, which manipulate and interpret the data stream in the sequence given by the connections between the nodes. Thereby, the researcher or interaction designer has a simple visual language to define the data flow, respectively the interaction. Squidy provides predefined nodes for all supported input and output devices and a selection of common filters such as a Kalman filter which reduces data noise and estimates the movement path based on the previous measured discrete contact positions. Thus, different fingers can be distinguished and tracked over time. The resulting movement path for each finger can then be individually analyzed in continuative filters identifying dwelling, selection or dragging gestures. This general approach is useful for all kinds of multitouch surfaces. Without Squidy, designers would have to implement the gesture tracking for every combination of input device and toolkit they want to use. With Squidy, they can reuse the gesture tracking for different devices even in parallel by connecting multiple input devices (e.g. iPhone & Touchlib) to the respective filter node.

Obviously, due to different sensing sizes, tracking resolutions and the environmental conditions, the filter parameters have to be adjusted to the corresponding devices in most cases. The determination of appropriate parameter values like a system noise level for the Kalman filter, however, is a highly iterative process and can be very time-consuming. To overcome that inefficiency, Squidy supports fast iterations and a direct comparison of different parameter values by facilitating the interactive manipulation of the data flow graph and the filter parameters at runtime. Using the concept of semantic zooming the nodes reveal additional information when zooming into the node such as the adjustable parameters (see Figure 2b), the description of the zoomed filter, current debug information and even its source code. The designer is able to access and modify the filter directly without losing the context, interrupting the tracking or starting up additional applications. The interaction library Squidy assists interaction designers and researchers in an effort of engineering novel user interface applications without knowledge of the API for input and output devices as well as complex filter algorithms.

ZOIL UI FRAMEWORK

Implementing a ZOIL user interface prototype without tool support is a challenging task. While the presentation and manipulation of objects on a digital canvas is a standard scenario for contemporary UI toolkits, a fluid semantic zoom into an information landscape containing hundreds and thousands of nested visible objects is much more demanding, especially if the objects contain high resolution photos, videos or web pages.

The ZOIL UI framework (ZF) provides the necessary software components for a rapid implementation of a ZOIL prototype with the Windows Presentation Foundation (WPF), C# and XAML. Internally ZF uses WPF's visual tree and scale transformations to create a zoomable hierarchy of nested UI elements. This hierarchy is rooted in the ZOIL information landscape component, which provides basic ZUI functions and also receives and processes Squidy events for ZUI navigation and Direct Manipulation. Each node in the zoomable hierarchy can contain arbitrary combinations and numbers of WPF UI controls including media elements or complex forms with data bindings to databases or web services. The zoomable hierarchy is typically defined by the designer using the declarative XML-based user interface markup language XAML which is a part of WPF. By extending XAML with a set of specialized ZOIL properties and components, even complex ZUI behavior such as different representations at different zoom levels or making standard UI controls zoom to fit the screen size after tapping can be specified without writing C# code. Furthermore ZF provides XAML templates which can be assigned to all kinds of controls to change visual styles or to make controls of a certain type dragable, sizeable or rotatable. Therefore even more elaborate ZOIL prototypes can be created without coding.

CONCLUSION & OUTLOOK

Based on our experiences and theoretical discussions, we believe that the design of SC has to be guided by theoretical frameworks such as EI and RBI and novel user interface paradigms such as ZOIL. The full potential of SC can only be unleashed by questioning and abandoning traditional WIMP concepts. This will result in a wider design space, but also in greater technological challenges, which can be met by providing tool support for iterative prototyping. Our future research work will be focused on refining the ZOIL paradigm and strengthen such tool support by building and testing high-fidelity tabletop prototypes for libraries, museums and automotive showrooms.

ACKNOWLEDGEMENTS

We would like to thank Roman Rädle and Andreas Engl for their outstanding contributions to Squidy and ZOIL. This work is supported by DFG GK-1042 "Explorative Analysis and Visualization of Large Information Spaces" and the project "Interactive Visualization for Gigapixel Displays" supported by the "Information Technology Baden-Württemberg (BW-FIT)" program.

REFERENCES

1. G. Casiez, D. Vogel, R. Balakrishnan, and A. Cockburn. The impact of control-display gain on user performance in pointing tasks. Human-Computer Interaction, 23(3):215–250, 2008.
2. D. Collins. Designing object-oriented user interfaces. Benjamin Cummings, Redwood City, CA, 1995.
3. P. Dourish. Where the action is: the foundations of embodied interaction. MIT Press, Cambridge, Mass.; London, 2004.
4. S. Foehrenbach, W. A. König, J. Gerken, and H. Reiterer. Natural interaction with hand gestures and tactile feedback for large, high-res displays. In MITH'08: Workshop on Multimodal Interaction Through Haptic Feedback, held in conjunction with AVI'08, 2008.
5. J. Gerken. Orientierung und Navigation in zoombaren Benutzungsschnittstellen unter besonderer Berücksichtigung kognitions-psychologischer Erkenntnisse. Master Thesis, University of Konstanz, 2006.
6. R. Harper, T. Rodden, Y. Rogers, and A. Sellen. Being Human: Human-Computer Interaction in the year 2020. Microsoft Research Ltd., Cambridge, England, 2008.
7. S. Herholz, L. L. Chuang, T. G. Tanner, H. H. Bülthoff, and R. W. Fleming. Libgaze: Real-time gaze-tracking of freely moving observers for wall-sized displays. Proceedings of VMV'08, 2008.
8. E. L. Hutchins, J. D. Hollan, and D. A. Norman. Direct Manipulation Interfaces, pages 87–124. User Centered System Design: New Perspectives on Human- Computer Interaction. Erlbaum, Hillsdale, NJ, 1986.
9. R. J. K. Jacob, A. Girouard, L. M. Hirshfield, M. S. Horn, O. Shaer, E. T. Solovey, and J. Zigelbaum. Reality-based interaction: a framework for post-wimp interfaces. In CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems, pages 201– 210, 2008.
10. H.-C. Jetter. Informationsarchitektur und Informationsvisualisierung für die Post-Wimp Ära, Master Thesis, University of Konstanz, 2007.
11. H.-C. Jetter, A. Engl, S. Schubert, and H. Reiterer. Zooming not zapping: Demonstrating the zoil user interface paradigm for itv applications. In Adjunct Proceedings of European Interactive TV Conference, 2008.
12. H.-C. Jetter, J. Gerken, W. A. König, C. Grün and H. Reiterer. HyperGrid - Accessing Complex Information Spaces. HCI 2005: People and Computers XIX - The Bigger Picture, 2005.
13. H.-C. Jetter, W. A. König, J. Gerken, and H. Reiterer. Zoil - a cross-platform user interface paradigm for personal information management. In Personal Information Management 2008: The disappearing desktop (a CHI 2008 Workshop), 2008. Video: http://hci.uni-konstanz.de/permaedia/permaedia.avi
14. P. Ravasio and V. Tscherter. Users' Theories of the Desktop Metaphor, or Why We Should Seek Metaphor-Free Interfaces, pages 265–294. Beyond the desktop metaphor: designing integrated digital work environments. MIT Press, Cambridge, Mass., 2007.
15. W. A. König. Referenzmodell und Machbarkeitsstudie für ein neues Zoomable User Interface Paradigma. Master Thesis, University of Konstanz, 2006.
16. W. A. König, H.-J. Bieg, T. Schmidt, and H. Reiterer. Position-independent interaction for large high-resolution displays. In IHCI'07: Proceedings of IADIS International Conference on Interfaces and Human Computer Interaction 2007, pages 117–125. IADIS Press, 2007.
17. T. Mandel. Windows vs. OS2, the GUI-OOUI war: the designer's guide to human-computer interfaces. Van Nostrand Reinhold, New York, 1994.
18. K. Perlin and D. Fox. Pad: an alternative approach to the computer interface. In SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques, pages 57–64, New York, NY, USA, 1993. ACM Press.
19. V. L. Wallace. The semantics of graphic input devices. SIGGRAPH Comput. Graph., 10(1):61–65, 1976.