Toward a Software Model and a Specification Language for Next-Generation User Interfaces

Orit Shaer, Robert J.K Jacob
Computer Science Department, Tufts University
161 College Ave. Medford, MA, USA 02155
{oshaer, jacob}@cs.tufts.edu

ABSTRACT
As user interfaces evolve from traditional WIMP to 'reality based interfaces', developers are faced with a set of new challenges which are not addressed by current software tools and models. This paper discusses these challenges and presents a software model and specification language which are designed to simplify the development of non-WIMP interfaces such as virtual environments. Finally, we discuss our plans to extend this model to support physical interaction and dynamic adaptation of interfaces.

INTRODUCTION

Current HCI research covers a diverse range of new interaction styles, including augmented reality, multi-modal interfaces, tangible user interfaces, physical user interfaces and lightweight interaction. Common to all of these is the change in interaction with computers, from a segregated and specialized activity to one more closely related to the real physical world. Thus, we identify these interaction styles as an emerging next generation of user interfaces and refer to them as reality-based interfaces.

Challenges for User Interface Software Tools

By leveraging experience, knowledge and skills that users already possess, reality-based interfaces offer the promise of interfaces that are easier to use. However, these interfaces are currently more difficult to build than traditional WIMP interfaces. To explore why current user interface software tools are not applicable to reality-based interfaces, consider some of the characteristics of current WIMP interfaces vs. next-generation interfaces as they are specified in table 1.

Many of the characteristic of WIMP interfaces such as single thread, discrete and precise tokens as well as explicit commands correspond to the characteristics of non-interactive programming languages processed by compilers. Indeed, many of the current user interface tools are built around compiler technology. As next-generation interfaces violate these assumptions, they are not well served by compiler-based approaches.

A recent practice in WIMP interfaces has been to codify available interaction techniques into a toolkit which contains a standard set of widgets. However, next-generation interfaces introduce a far richer range of interaction techniques and interactive behaviors and while the effort to codify existing interaction techniques has begun [2], the search for new interaction techniques and support technologies continues. Thus, software tools which are easily extended to support new interaction techniques and sensing technologies are needed.

Current	Next Generation
Single-thread dialogue	Concurrent, asynchronous dialogues
Discrete and precise tokens	Continuous and discrete inputs and responses, probabilistic input
Sequence, not time, is meaningful	Real-time requirement, deadline-based computations
Explicit commands	Intentional and passive interaction
Consists of predefined interactive objects and behaviors.	Introduces new types of objects with new types of interactive behavior.
Single output channel	Parallel digital and physical output channels
Standard IO hardware	Employ a variety of devices and sensing technologies
Operate in a static and isolated computational environment	Operate in dynamic, ad-hoc, heterogeneous environments

Table 1. The characteristics of current vs. next-generation UI.

To date, WIMP interfaces primarily operated in the homogeneous and computationally isolated desktop environment. Next-generation interfaces will operate in dynamic heterogeneous environments where a large variety of devices communicate and share resources such as storage and interaction capabilities. These interfaces may spread across multiple devices which are temporarily connected. As devices join and leave a computational environment, changing the available landscape of resources, user interfaces should be constantly adapted to employ available interaction devices. User interface software tools will thus need to provide ways to build interfaces given uncertainty about the devices they will employ and supply services such as dynamic adaptation of user interfaces.

Though existing techniques could be extended in ad-hoc ways to address some of the aspects unique to next-generation UI's, they would fall short of producing a comprehensible specification at a high level abstraction. The approach of User Interface Management System (UIMS) [3] seems like a solid foundation for the development of software tools for next-generation user interfaces. However, to date, few UIMS's address the unique characteristics of next-generation interfaces.

Our goal is to develop a new model, a language and a UIMS for describing and implementing reality-based interfaces. The language will enable to: introduce new objects and interactive behaviors, design a reality-based interface given the uncertainty of the devices it may employ, directly specify concurrent and interrelated dialogues as well as parallel, continuous and discrete interaction. The next sections discuss a model and a language for describing and implementing non-WIMP interfaces and our plans to extend it to support reality-based interfaces.

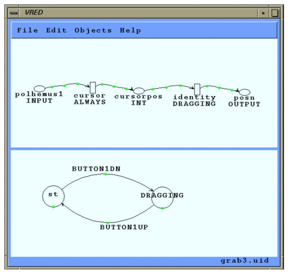
A SOFTWARE MODEL AND A SPECIFICATION LANGUAGE FOR NON-WIMP USER INTERFACES

In a recent project, Jacob et al. presented a software model, a language and a development environment for describing and programming the fine-grained aspects of non-WIMP user interfaces, such as virtual environments [1].

Their approach is based on the view that the essence of a non-WIMP dialogue is a set of continuous relationships—most of which are temporary. The model addresses the continuous aspect of non-WIMP interaction explicitly by combining a data-flow component for describing continuous relationships with an event-based component for specifying discrete interactions. The event-based component can enable or disable individual continuous relationships. The model also supports parallel interaction implicitly because it is simply a declarative specification of a set of relationships that are in principle maintained simultaneously. The key ingredient of this model is hence the separation of non-WIMP interaction into two communicating components. The model abstracts away many of the details of specific interaction devices and treats them only in terms of the discrete events they produce and the continuous values they provide. It thus reduces the effort required to integrate novel devices into virtual reality applications. Based on this model, a user interface description language was developed and implemented in visual and textual (SGML-based) forms. Figure 1 shows the specification of a common interaction in virtual reality, grabbing and dragging an object using graphical notation. The upper half of the screen shows the specification of the continuous interaction using a dataflow diagram. The lower half shows the event handler in the form of a state diagram.

PMIW: A User Interface Management System for non-WIMP Interactions

To demonstrate that these model and language do not compromise real-time performance, PMIW, a UIMS that implements these new model and language was presented. The PMIW software is written in C++ for Unix while the Silicon Graphics Performer software is used for graphics and rendering. Each form of the language is ultimately translated into C++ code, which runs on the PMIW user interface software. The PMIW main loop: reads input from an X window and the interaction devices, dispatches the inputs (either to the data flow component or to the event handler), recalculate the data flow component, propagates update the scene graph data, reads the head tracker, and then render the scene graph. Figure 2 shows the 'grabbing and dragging' specification running in the PMIW system. The diamond-shaped cursor is permanently attached to the user's hand; the other object can be grabbed and moved.



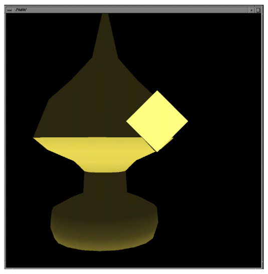


Figure 1 & 2. Specification and rendering of 'grabbing and dragging' interaction.

TOWARD A UIMS FOR REALITY-BASED INTERFACES

We intend to extend the PMIW model, language and software to support unique features of reality-based interfaces such as physical interaction, concurrent dialogues as well as combined intentional and passive interaction. To accomplish this goal, we began with identifying a set of core constructs sufficient for describing tangible user interfaces (TUIs). This set of constructs is introduced in the Token and Constraint (TAC) paradigm [5]. The TAC paradigm approach is based on describing a TUI as a set of relationships between two types of physical objects: *tokens* which represent digital information and *constraints* which provide the context for token manipulation. The relationship between a *token* and a set of *constraints* is called a *TAC*. Similar to widgets, TAC objects encapsulate the set of manipulation actions that can be performed upon a physical object in a TUI. Our initial hypothesis is that the essence of reality-based interaction is a set of concurrent relationships among interaction objects that encapsulate a set of meaningful interaction actions (continuous or discrete) for a given context. These relationships might be engaged by intentional or passive interactions. Building upon this approach, we are currently developing a specification language and a software model. Furthermore, we intend to extend the PMIW software to support dynamic adaptation of reality-based interfaces to a changing landscape of interaction resources.

REFERENCES

1. Jacob, R.J.K., L. Deligiannidis, and S. Morrison, *A Software Model and Specification Language for Non-Wimp User Interfaces.* ACM Transactions on Computer-Human Interaction (TOCHI), 1999. **6**(1): p. 1-46.
2. Klemmer, S.R. and J.A. Landay. *Papier-Mache: Toolkit Support for Tangible Input*. in *CHI2004*. 2004.
3. Olsen, D.R., *User Interface Management Systems: Models and Algorithm,* Morgan Kaufman, 1992.
4. Shaer, O., N. Leland, E.H Calvillo, and R.J.K. Jacob, *The TAC Paradigm: Specifying Tangible User Interface.* Personal and Ubiquitous Computing 8(5), 2004.