A Specification Paradigm for Design and Implementation of Non-WIMP User Interfaces

Stephen A. Morrison
Department of Electrical Engineering
and Computer Science
Tufts University
161 College Avenue
Medford, MA 02155-5528
(617) 627-3217
smorrisn@eecs.tufts.edu

Robert J. K. Jacob
Department of Electrical Engineering
and Computer Science
Tufts University
161 College Avenue
Medford, MA 02155-5528
(617) 627-3217
jacob@eecs.tufts.edu

ABSTRACT
The SHADOW System is a user interface management system designed to address the specific needs of non-WIMP interfaces such as virtual environments, gesture recognizers and other interactions that involve highly parallel, continuous interaction. The proposed UIMS consists of a graphical specification language based on augmented transition networks and data flow graphs, a code translation system which supports dynamic constraint binding, modular design and code reuse, and a run time engine designed to optimize the use of processing resources within a time sensitive environment while preserving a layer of platform independence for the application.

Keywords
Constraint programming, interface specification, non-WIMP, SHADOW, software engineering, state transition diagram, user interface description language (UIDL), user interface management system (UIMS), virtual reality (VR), visual programming.

INTRODUCTION
Most current user interface specification languages and toolkits are based on serial, discrete, token exchange paradigms which, in general, perform an acceptable job of implementing traditional WIMP (Window, Icon, Menu, Pointer) interfaces commonly found in todays' office automation software. Unfortunately, these tools are ill suited to address the needs of emerging non-WIMP interaction styles such as virtual environments. This limitation stems from the general characteristics of non-WIMP user interfaces. These emerging interaction styles commonly rely upon: full duplex, asynchronous, interrelated dialogues; a blend of continuous and discrete inputs and responses; and, implicit commands and probabilistic input events. Additionally, some forms of non-WIMP interactions, such as immersive virtual reality, must also contend with real time processing constraints and deadline-based computations [2, 3].

The lack of applicable software tools has forced many interface developers to resort to using ad-hoc, low-level programming approaches when dealing with non-WIMP system. These approaches are usually adequate for the task at hand from a functional standpoint but drastically add to the complexity of the development while hindering efforts at code reuse, platform independence, and long term maintenance.

The Specification Problem
From a designer's perspective, a system of techniques and abstractions needs to be developed which allows both the behavior and semantic meaning of all interface elements to be clearly defined in a reusable fashion. Such a specification should allow conceptual continuity between our cognitive understanding of an object or phenomenon and our description of it. Thus, an action which a typical user would percieve to be a discrete event, such as a mouse click, can be handled as a singular event token while a continuous force, such as gravity, may be described as a permanent constraint effecting all object with mass.

The lack of standards and emerging nature of the domain of non-WIMP interfaces further complicates the specification problem in that new input and output devices are constantly being introduced as are novel interaction techniques themselves. Any tool or language seeking to service this domain must be extensible or risk rapid obsolescence both for the tool and any system developed with it.

Many proposed solutions to the specification problem offered to date have explored many of the conceptual issues described above on a small scale [1,3,4] but have done little to address the problems of scale which arise when trying to specify an entire interface rather than the individual behavior of an element within an interface. As interfaces grow in size and complexity, support for good software engineering practices such as modular design and traceability become vital to the success of the system.

Implementation Issues

In addition to the cognitive issues of describing semantic behavior, non-WIMP systems must also deal with very practical issues of performance, portability, and maintainability. Any UIMS targeted at a non-WIMP domain needs to be sensitive to these issues and should provide mechanisms which allow run time performance criteria and deadline contingency plans to be specified in a manner which is easily discernible and platform independent.

A LANGUAGE MODEL

The SHADOW System seeks to address these concerns by providing a graphical specification language consisting of a data flow graph and an augmented transition network and is based on the PMIW model proposed by Jacob [3]. This language is highly declarative in nature, supports loosely coupled, modular design and relies upon a run time engine to resolve constraints and to manage conceptually parallel tasks within uniprocessing environments.

The data flow graph consists of a network of links and variables. Links are conceptually continuous, modular data transforms or user defined I/O channels which can be selectively activated or deactivated in response to discrete tokens or performance restrictions. Variables serve as data repositories and conduits. By selectively controlling the topology of the data flow graph, the designer may specify both permanent behaviors (such as the force of gravity) as well as temporary relationships (such as the location of an object with respect to one's hand while being carried).

The augmented transition network is designed for servicing discrete event tokens (raised by links) and uses these tokens to dynamically alter the topology of the data flow graph based on system status information flags. In this fashion, both discrete events and parallel, continuous relationships may be modelled in a way that is both segregated and interrelated.

UIMS RUN-TIME CONSIDERATIONS

The SHADOW System run time engine provides the infrastructure which allows the interface designer to address conceptual and semantic issues of design without becoming bogged down in the details of implementation. The engine is responsible for internal task management, event propagation, and constraint binding. Additionally, the engine provides facilities which allow the designer to statically specify both real time performance criteria and contingency plans to help the system automatically adjust processing loads to meet those criteria should CPU processing time consumption become a problem.

CURRENT WORK

To date, the SHADOW System consists of the SHADOW-Talk visual language specification, the SHADOW-Script text language, an initial version of the code translator used to generate C++ from SHADOW-Script specifications, and a prototype run time engine. A visual editor for the SHADOW-TALK language is under development and will be added to the UIMS to complete the graphical programming environment.

As an on going effort, the SHADOW System is being applied to a variety of tasks to explore and define the limits of its ability to meet the needs of non-WIMP interface designers. Areas under investigation include: automated support for decimation policies; event abstraction; gesture recognition and other probabilistic inputs, generic constraint specification and physical law simulation; and, support for large scale development.

ACKNOWLEDGEMENTS

This work was supported by National Science Foundation Grant IRI-9625573, Office of Naval Research Grant N00014-95-1-1099, abd Naval Research Laboratory Grant N00014-95-1-G014. Additionally, I would like to thank Quan Lin and Leonidas Deligiannidis for their ideas and suggestions regarding the implementation of the SHADOW UIMS itself

REFERENCES

1. Abowd, G. D. and Dix, A. J., "Integrating Status and Event Phenomena in Formal Specifications of Interactive Systems," *Proceedings of the ACM SIGSOFT'94 Symposium on Foundations of Software Engineering*, 1994

2. Green, M. and Jacob R. J. K., "Software Architectures and Metaphors for Non-WIMP User Interfaces," *Computer Graphics*, vol 25, no. 3, pp 229-235, July 1991.

3. Jacob, R. J. K., "A Visual Language for Non-WIMP Interfaces," ***Proceedings IEEE Symposium on Visual Languages*****, pp. 231-238, IEEE Computer Society Press (1996).**

4. Jacob, R. J. K., "A Specification Language for Direct Manipulation User Interfaces," *ACM Transactions on Graphics*, vol. 5, no. 4 pp 238-317, 1986.

5. Newman, W. M., "A System for Interactive Graphical Programming," *Proceedings Spring Joint Computer Conference*, pp 47-54, AFIPS, 1968.