Task-Oriented Representation of Asynchronous Interfaces

Antonio Siochi
H. Rex Hartson

TR 88-42

CHI'89 PROCEEDINGS, ACM, New York, NY

MAY 1989

TASK-ORIENTED REPRESENTATION OF ASYNCHRONOUS USER INTERFACES

Antonio C. Siochi
H. Rex Hartson

Department of Computer Science
Virginia Tech
Blacksburg, VA 24061

ABSTRACT

A simple, task-oriented notation for describing user actions in asynchronous user interfaces is introduced. This User Action Notation (UAN) allows the easy association of actions with feedback and system state changes as part of a set of asynchronous interface design techniques, by avoiding the verbosity and potential vagueness of prose. Use within an actual design and implementation project showed the UAN to be expressive, concise, and highly readable because of its simplicity. The task- and user-oriented techniques are naturally asynchronous and a good match for object-oriented implementation. Levels of abstraction are readily applied to allow definition of primitive tasks for sharing and reusability and to allow hiding of details for chunking. The UAN provides a critical articulation point, bridging the gap between the task viewpoint of the behavioral domain and the event-driven nature of the object-oriented implementational domain. The potential for UAN task description analysis may address some of the difficulties in developing asynchronous interfaces.

KEYWORDS: Notation, interface design representation, asynchronous user interfaces, task-orientation, user actions, task description analysis.

INTRODUCTION

The dynamic nature of user interfaces has always been difficult to represent. A good notation reduces the semantic gap between entities in the designer's mind and objects with which that designer must communicate the design. Such entities lie in two interaction metaphors postulated by Hutchins, Hollan and Norman [8] – the conversational and the model world. The first is sequential in nature (as in command line interfaces) while the second is asynchronous (as in direct manipulation interfaces).

There are many techniques for explicitly representing the control flow of sequential dialogue, including BNF and state transition diagrams. However, representational needs for asynchronous interaction are difficult to satisfy with such techniques [10]. For example, how are the user actions required in a direct manipulation [17] interface specified? How are multi-thread and concurrent dialogues represented?

This paper introduces a task-oriented representation technique for asynchronous interaction and focuses on a notation called the User Action Notation (UAN) for representing user action sequences involved in the execution of a task. *The UAN is part of a set of techniques that, taken together, is used to describe interface designs.* This set of techniques has been successfully used to design the interface of a UIMS, where a direct manipulation interface was required. Because the project involved designers, implementors, and evaluators, communication of designs was vitally important. Our initial designs were in prose, supplemented with illustrations. These were time-consuming and varied in expressiveness. The descriptions were often verbose, imprecise, and typically difficult to read, understand, and change.

Our need was for an expressive and concise notation specifying user actions in relation to screen objects, as well as techniques for associating feedback and state changes with those actions. Because the design process is driven by requirements and task analysis, descriptions had to be expressible primarily from the viewpoint of the user, not the computer. Moreover, the interface's asynchronous nature demanded a technique which avoided explicit specification of control flow among tasks. The purpose of the UAN is therefore to communicate with all developer roles the user actions required to perform a task in an asynchronous interface.

The next section summarizes work related to the problem. The rest of the paper presents the UAN by describing portions of a well-known interface, then discusses some UAN characteristics.

RELATED WORK

Models and meta-languages for interfaces exist which can describe the task structure of an interface. They are formal in nature since their creators intended to use them

pp. 183–188