Naval Research Laboratory
Washington, DC 20375-5320

NRL/MR/5530--94-7488

Use of the User Action Notation at the Naval Research Laboratory Human-Computer Interaction Laboratory

Joe Chase
Deborah Hix
David Tate
James Templeman

Human-Computer Interaction Laboratory
Information Technology Division

June 30, 1994

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave Blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 30, 1994	Interim

4. TITLE AND SUBTITLE

Use of the User Action Notation at the Naval Research Laboratory Human-Computer Interaction Laboratory

5. FUNDING NUMBERS

PE - 62234N

6. AUTHOR(S)

Joe Chase, Deborah Hix, David Tate, and James Templeman

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
Washington, DC 20375-5320

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/MR/5530--94-7488

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report describes introduction of researchers in the Information Technology Division of the Naval Research Laboratory (NRL) to the User Action Notation (UAN), a notation developed at Virginia Tech for representing the design of the interaction component of interactive systems. This report also presents use of the UAN in describing a variety of unique, innovative interaction techniques to evaluate the notation for its ability to represent such techniques, and exploration of possibilities for future research and technology transition with the UAN collaboratively between NRL and Virginia Tech.

14. SUBJECT TERMS

Human-computer interaction
User interface design
User action notation
Methodology

15. NUMBER OF PAGES

35

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std 239-18
298-102

USE OF THE USER ACTION NOTATION AT THE NAVAL RESEARCH LABORATORY HUMAN-COMPUTER INTERACTION LABORATORY

Purpose of Visit

Mr. Joe Chase, a PhD Candidate in the Department of Computer Science at Virginia Tech, spent two weeks during Summer 1993 (18 - 30 July) in the HCI Laboratory of the Naval Research Laboratory. The purpose of this visit was threefold: to introduce researchers in the Information Technology Division of the Naval Research Laboratory (NRL) to the User Action Notation (UAN), a notation for representing the design of the interaction component of interactive software systems; to use the UAN in describing a variety of unique, innovative interaction techniques to evaluate the notation for its ability to represent such techniques; and to explore possibilities for future research and technology transition with the UAN collaboratively between NRL and Virginia Tech.

The first of these goals, introducing NRL researchers to the UAN, was accomplished in three ways. An overview introduction to the UAN as a notation and a method for interaction development was presented on 7/27/93. This presentation was followed by a 2 hour discussion, including approximately a dozen people, about the UAN and its application. The draft version of a UAN tutorial was described as a means for researchers to reference information about the UAN. A UAN description for an existing system at NRL, the Damage Control Information System (DCIS), was developed to provide an example of the potential role of the UAN in the analysis of existing interfaces as well as for future design.

The second goal, evaluating the UAN, especially with respect to innovative interaction techniques, was accomplished in two ways. The DCIS system was described using the UAN. This provided a sample UAN description for future reference. UAN descriptions of the basic task of using a device for a variety of the unique interaction techniques available at NRL were also developed.

The third goal of this visit was to identify areas of future research and technology transfer between NRL and Virginia Tech. NRL provides a unique opportunity for human-computer interaction research and application because of its focus on the transfer of technology and ideas from academia to application. Being able to observe and interact with researchers at NRL, as well as being able to experiment with new interaction techniques, has allowed us to identify a number of outstanding issues which require further study. These issues include, but are not limited to, the relationship between the UAN and virtual reality devices and the examination of device vocabularies as a way of approaching new interaction techniques.

The User Action Notation

The User Action Notation (UAN) is a user- and task-oriented notation that describes the behavior of a user and an interface during their cooperative performance of a task (1). The primary abstraction of the UAN is a *user task* – a user action or group of temporally related user actions performed to achieve a work goal. A user interface is represented as a quasi-hierarchical structure of asynchronous tasks, sequencing within each task is independent of that in the others. User actions, corresponding interface feedback, and state information are presented at the lowest level. Levels of abstraction, where lower level tasks are combined under a single general task name, are used to hide these details and represent the entire interface. At all levels, user actions and user tasks are ordered and combined using temporal relations such as sequencing, interleaving, and concurrency. Since textual notations are not always convenient for specifying all components of an interface, the UAN includes screen

Manuscript approved March 31, 1994.

pictures, or scenarios, and can be supplemented with state transition diagrams to indicate precisely how the user interacts with the interface. The following example shows the UAN's hierarchical and temporal approach to the description of a user interface.

As an example, consider a simple calendar management system. We assume systems and requirements analysis have determined that the user wants to perform five basic tasks: viewing, adding, modifying, and deleting an appointment on the calendar, and setting an alarm associated with a given appointment. Our first step would be to specify the hierarchical relationships at this highest level (Figure 1).

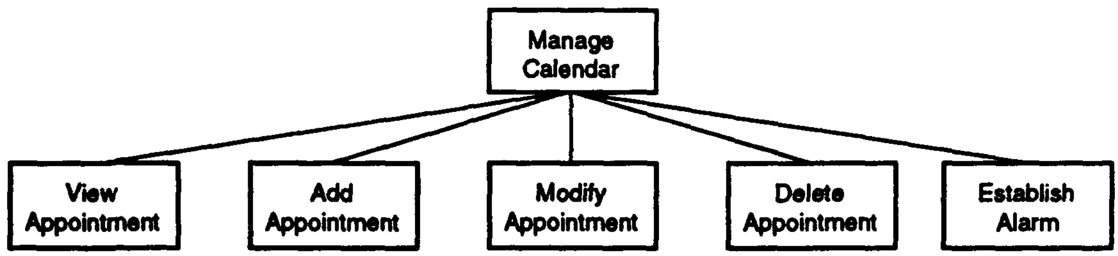


Figure 1. Highest level of hierarchy for calendar management system

However, having specified the hierarchical decomposition of tasks, we must now specify temporal relationships among these user tasks (Figure 2).

Task: manage calendar		
User Action	**Feedback**	**Interface State**
OR(add appointment, view appointment, modify appointment, delete appointment, set alarm)*		

Figure 2. Highest level of abstraction for calendar management system

In Figure 2, OR indicates that the user may choose any one of the five tasks that follow. The asterisk (*) after the disjunction indicates that the user may perform this choice zero or more times. (We borrow this notation from the Kleene star closure operator of formal language theory; the UAN also provides a plus (+) operator to indicate that an action must be performed one or more times.) Thus, the UAN description in Figure 2 specifies that the user can perform a sequence of tasks of any length (including zero), with each task selected independently from those specified in the disjunction. Each of these tasks could then be decomposed further using the same process.

The UAN is primarily a notation for behavioral representation of an interaction design. However, through empirical work with industrial users of the UAN, we have found that it has a variety of uses across the entire interaction development process. As we continue to collect information on how the UAN is being used, a composite, seamless method of organization, representation, and communication has emerged. This method and the basic symbols of the UAN are fully described in a UAN Tutorial (2)., and will not be detailed in this document. However, for reference, Appendix 2 contains a summary of the most frequently used UAN symbols. For the reader who is completely unfamiliar with UAN, we recommend obtaining the UAN Tutorial from Virginia Tech.

We have found the UAN to be useful both during the design process and as a reverse engineering tool for existing user interfaces. The method of application of the UAN is very similar in both cases as will be described in more detail in the next section.

A UAN Example: The Damage Control Information System

A Damage Control Information System (DCIS) has been developed in the Information Technology Division at NRL. DCIS provides the user with the ability to monitor (and in some cases control) various damage control apparati throughout a ship from one central location. This system monitors smoke, heat, flood, and flame detectors, and allows monitoring and control of alarms, fire main valves, fire main pressure gauges, and fire pumps. The user interface to DCIS provides a user with direct manipulation control over a representation of the physical object which they are trying to manage.

The purpose of writing a complete UAN description of DCIS was not to critique its user interface but rather to evaluate the abilities of the UAN to describe this type of interface and to provide an example for future reference. The UAN provides a view of the interaction which helps to point out aspects of the interaction design that may have previously gone unnoticed. The UAN typically is used as a design representation technique as an interactive system is being designed. The process of developing UAN descriptions is similar whether they are being written for a new user interface or for an existing interface.

The first step in the process of applying the UAN to an existing interface design is to develop a hierarchical decomposition of tasks in the user's problem domain. This can be done either by interacting with the interface or prototype if it has been developed or through analysis of design documents if the interface is still in the design phase. For example, in the case of DCIS, the user's global task of managing damage control activities was decomposed into four tasks as shown in Figure 3. These tasks were identified through interaction with the existing DCIS prototype.

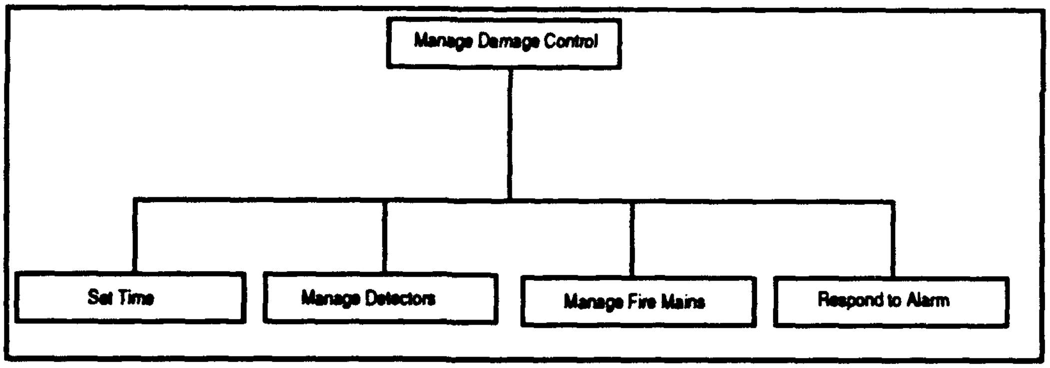


Figure 3. First level of decomposition of DCIS tasks

Each of these tasks can then be further decomposed until at some point, all the user's problem domain tasks have been decomposed as much as possible. A complete UAN description of the DCIS is given in Appendix 1. If we were designing a new system, we could have reached this point without making any implementation decisions. For a new system, it would be useful at this point to define the interaction platform, i.e., devices, buttons, techniques, etc. Whether for a new system or as in this case for an existing system, these interface objects and groups of objects can be represented by definitions that describe the objects and their behavior. Some examples of object definitions from DCIS are shown in Figure 4.

Definitions:

Class: buttons
Description: objects appearing on the screen that look three dimensional
Highlighting: buttons appear two dimensional (!)

Group: toggle buttons
Description: bi-state button that change state on selection
Highlighting: buttons show inverse video (!). Button text shows inverse state (!!).
Members: AM/PM

Group: default buttons
Description: rectangular buttons that control default configuration of fire mains
Highlighting: buttons become slightly darker on mouse down (!)
Members: Xray, Yoke, Zebra

Figure 4. Sample object definitions from DCIS

These object definitions then lead us to develop articulation-level macros that operate on these objects (e.g., **select button**). The articulation level is the level in a UAN description at which the user is actually using an input device to accomplish some task. This is the level at which inconsistencies in user interaction design are discovered. This is done by creating, either from scratch or through examination of an existing interface, generic macros such as **select**, etc. If we discover that similar objects have, for example, different behavior for the same user actions, or the same behavior for different user actions, then we have discovered inconsistency. Figure 5 shows an example from DCIS where two similar objects are behaving very differently under the select task.

Task: Select Pressure Gauge(pressure gauge)		
Arena: multiple		
User Action	Feedback/ Presentation	Interface State
~[pressure gauge icon] Mv	pressure gauge icon!	selected = pressure gauge activated = pressure gauge
M^		

Task: Select Detector(button)			
Arena: multiple			
User Action	Non-Mainline Action	Feedback/ Presentation	Interface State
~[button icon] Mv		button icon!	selected = button
	~[x,y] not in button icon	button icon-!	
	~[button icon]	button icon!	
M^		button icon-!	activated = button
	~[x,y] not in button icon	button icon-!	
	M^		activated = none selected = none

Figure 5. Sample select tasks from DCIS showing behavioral inconsistencies

To briefly explain the UAN notation shown in these examples, in the first cell of the User Action column of the first example, ~[pressure gauge icon] means "move the cursor (~) to the pressure gauge icon and depress the mouse button (Mv)". In the Feedback/Presentation column associated with this user action, pressure gauge icon! means highlight (!) the pressure gauge icon. The Interface State Column indicates that pressure gauge becomes part of a set named selected and also another set named activated at this point. Finally, in the second User Action cell, the mouse button is released (M^). In the second example, the Non-Mainline Action column indicates those user actions that can be performed that are not directly related to the primary task, here Select Detector.

These examples show that the user pressing the mouse button (Mv) creates different results for these two tasks. In the case of pressure gauge buttons, selection and activation both occur as a result of the user pressing the mouse button. However, in the case of detector buttons, selection occurs as the result of the user pressing the mouse and activation occurs as a result of the user releasing the mouse. While this is a simplistic example, all these little inconsistencies may combine to confuse a user and reduce productivity. In the process of writing this description by carefully performing each possible task in the DCIS, several similar inconsistencies were uncovered. This finding reinforced our previous findings that the UAN provides inherent consistency checking at the articulation level through the attempt to combine similar user actions and tasks into macros.

Exploring Alternative Interaction Techniques

One of the purposes of this visit, as noted above, was to use UAN to describe a variety of unique interaction techniques to evaluate the notation for completeness. NRL is a perfect setting for this since much of the work taking place centers around alternative interaction techniques such as the boom, egocentric projection, and eye gaze technology.

The first interaction technique to be examined was egocentric projection. This system allows the user to control their view of objects on the screen in three dimensions by merely moving their head. If they move their head closer to the screen, the image on the screen is magnified. If they move their head farther from the screen, the image on the screen is reduced in size. Further, the user can pan up and down or left and right on the screen by moving their head in the direction in which they wish the screen to pan. Currently, there is no capability to select objects in this technique. This is important since it greatly reduces the vocabulary for this device by eliminating selection, dragging, and activation.

In writing the UAN description of this task, it was difficult to decide whether the basic use of this device was made up of one task (e.g., use device) or two tasks (e.g., pan and zoom) combined to form one task. Originally, we wrote it as two tasks. However, after discussions with the developer of the system and several other researchers, it became apparent that from a user perspective, this should be one task. This is because a user does not think about moving in the (x,y) plane separately from the z axis when manipulating a 3-D image on the screen. The user will move as directly as possible to the point in three-space that accomplishes their intended purpose. Thus the task of using the egocentric projection technique would be written as follows:

Task: use egocentric projection		
Arena: simulator		
User Action	Feedback/ Presentation	Interface State
~(x,y,z)	redisplay view from (x,y,z)	

The second interaction technique to be examined was the Fake Space boom. The boom is a virtual reality device which allows a user to scan in any direction by turning their head and the mask of the boom in that direction. The user may move in any direction by either moving the boom in that direction for small movements or by using "fly" buttons–one on either handle–to move quickly forward or backward. As with egocentric projection, selection is not implemented with this technique. Thus the vocabulary for this device is relatively small.

Again, as with egocentric projection, the difficulty in representing this technique in UAN arose from the question of whether it should be represented by multiple tasks or by one task. On the surface, it would appear that use of this technique falls into one of three tasks: panning, walking, or flying. In fact our first representation of this task was made up of three tasks. However, again after discussions with other researchers and users of the technique, it became apparent that this was not the case. As with egocentric projection, the user will merely do whatever is necessary to move from where they are now to the point in three-space that accomplishes their purpose. Of course, with the boom, it is not only the position of the view within the three-space that is important, but also the orientation of the view. Therefore, representation of the task of using the boom would be written in UAN as follows, where CONCURRENT means perform the following actions simultaneously:

Task: use boom		
Arena: simulator		
User Action	**Feedback/ Presentation**	**Interface State**
CONCURRENT(~(x,y,z), orient(a,b,c))	redisplay view from (x,y,z) with orientation (a,b,c)	

The third interaction technique, eye gaze, resulted in a slightly different description. With this technique, a user is able to move the cursor on the screen by looking at the object or location on the screen where they wish the cursor to go. Objects are selected if the cursor is within their context. If the user holds the cursor on a particular object (i.e., gazes) for a preset time, then the object is activated. For example, if the user holds their gaze and thus the cursor on a menu heading for longer than a preset time n, then the menu will be activated. Thus the UAN description of the basic eye gaze task would actually be two tasks written as follows:

Task: select using eye gaze(object)		
Arena: simulator		
User Action	**Feedback/ Presentation**	**Interface State**
~[object icon]	object icon!	selected = object

Task: activate using eye gaze(object)		
Arena: simulator		
User Action	Feedback/ Presentation	Interface State
select (object)		
(t > n)		activated = object

Suggestions for Future Work

Other alternative interaction techniques and input devices, such as voice and gestural interfaces, were discussed. Several basic ideas came out of these discussions. First, it appears that any given input device has a vocabulary. This means that even though there may be an infinite number of possible physical actions a user may do with a device, there are a limited number of recognizable actions that translate into a user accomplishing a specific task with an interface. This vocabulary, once identified, is easily describable, as shown by the boom and egocentric projection examples. However, identifying the complete vocabulary of a given device may not be a trivial matter. For example, there are a number of actions a user can do with a mouse, such as gestures or triple click, which may be part of the vocabulary of the mouse for some applications and may not be part of it for others. The idea of collecting a library of UAN descriptions of articulation-level macros and tasks for the vocabulary of known devices is one that is interesting for future work.

Second, there was a concern that the examples shown above for the boom and egocentric projection seemed quite simplistic for such complicated devices. After further discussion, it became apparent that while these devices are technically quite complicated, task descriptions for them are simplistic because from the user's view–which the UAN captures–they are very simple devices to operate.

The third issue centered around a premise underlying the UAN, that it is not necessary or useful to represent physical user actions that result in virtual user actions in an interface. For example, the UAN represents moving the cursor on the screen but does not represent the user moving a hand or eye or whatever physical action caused the cursor to move. A number of possible methods of representing these physical actions were discussed. One idea was to extend the UAN to a physical layer below the articulation level where a user actions of the articulation level are feedback of the physical level. While this approach provides an intuitive solution and provides a certain symmetry, it does not appear to be sufficient. The problem is that there are a very large number of possible physical actions to accomplish a single virtual action. Thus this physical level, written in UAN or any similar notation, would be prohibitively large. A more practical solution is to create a library of interaction devices, their associated vocabulary, and physical movements that accomplish the actions in the vocabulary. In this way, time-motion studies could provide assessment tests for whether an individual user will be able to accomplish a given task with a particular device. Time-motion studies have already been done for five of the basic input devices available today: mouse, trackball, joystick, tablet, and cursor keys.

The fourth area of discussion centered on the method of representing continuous, or seemingly continuous, activity–either user or system–with UAN. Currently, the UAN employs the method used by state transition diagrams and other notations, which is to represent continuous activity as an iteration of discrete activities. While this is somewhat intellectually unsatisfying and over-simplified, it appears to be sufficient for our purposes of representation, since a computer models continuous activity in the same way.

Another area of discussion was that of the formality or informality of the UAN notation. The concern was raised that for use in a technical environment, the notation should be more formal and/or more structured–possibly even standardized–to support consistent communication among developers, and also to support automated analysis, tool development, etc. However, this contradicts our previous findings among industrial clients who have actually complained that the UAN is already too formal and that they would prefer a more natural language notation. We have purposely made the UAN completely open to allow its users to modify and extend it to meet the unique needs of their particular user interface development environment. We encourage them to adopt whatever notational style, content, and conventions they prefer. In this way, if a group of UAN users wishes to formalize their in-house use of the notation, they may do so, while other users may choose, for example, to substitute words for symbols to get closer to natural language. The issue of standardization versus open notation will continue to be investigated.

The final area of discussion, related to several of the previous ones, is developing a case-based "library" of UAN descriptions. Such a set of UAN "idioms" or "behavioral widgets" would particularly help address vocabulary issue and standardization issues. This would greatly facilitate writing UAN descriptions.

All the above issues could be fruitful topics for further collaborative work and technology transfer between NRL and Virginia Tech. NRL provides unique opportunities for such collaboration because of its collection of innovative interaction techniques, its highly skilled researchers, and its focus on technology transfer from academia to application.

Acknowledgements

This work was sponsored by the Decision Support Technology block (RL2C) within the ONR Exploratory Development Program, managed by Elizabeth Wald. The authors wish to acknowledge Dr. Helen Gigley and the HCI Laboratory for funding Mr. Chase's visit to NRL.

References

1. D. Hix and H. R. Hartson, *Developing User Interfaces: Ensuring Usability Through Product and Process.* John Wiley & Sons, Inc., New York, (1993).

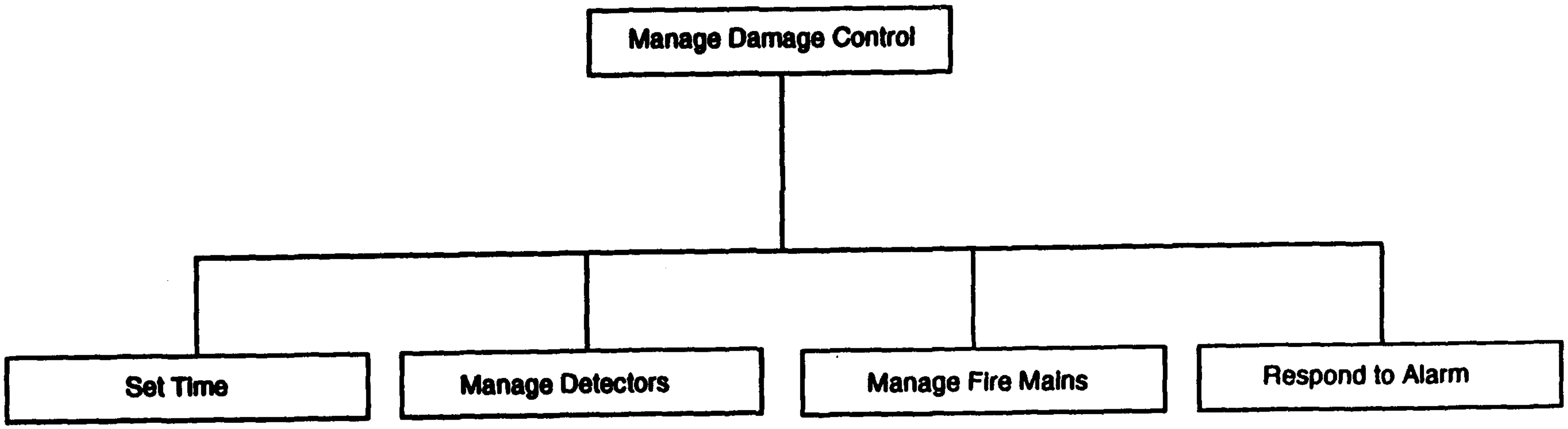
2. J. D. Chase, Jeffrey L Brandenburg, H. Rex Hartson, and Deborah Hix, *UAN Tutorial*, Department of Computer Science Technical Report, Virginia Tech, (1993).

APPENDIX 1

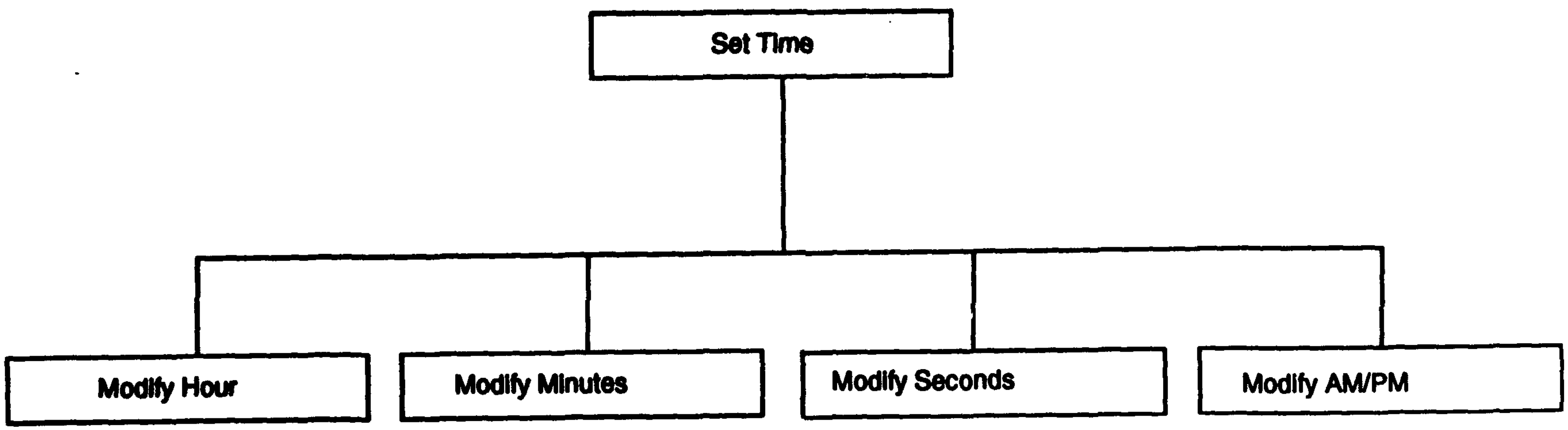
Complete UAN Description of the Damage Control Information System

NRL Damage Control Information System: UAN Description

Level 1

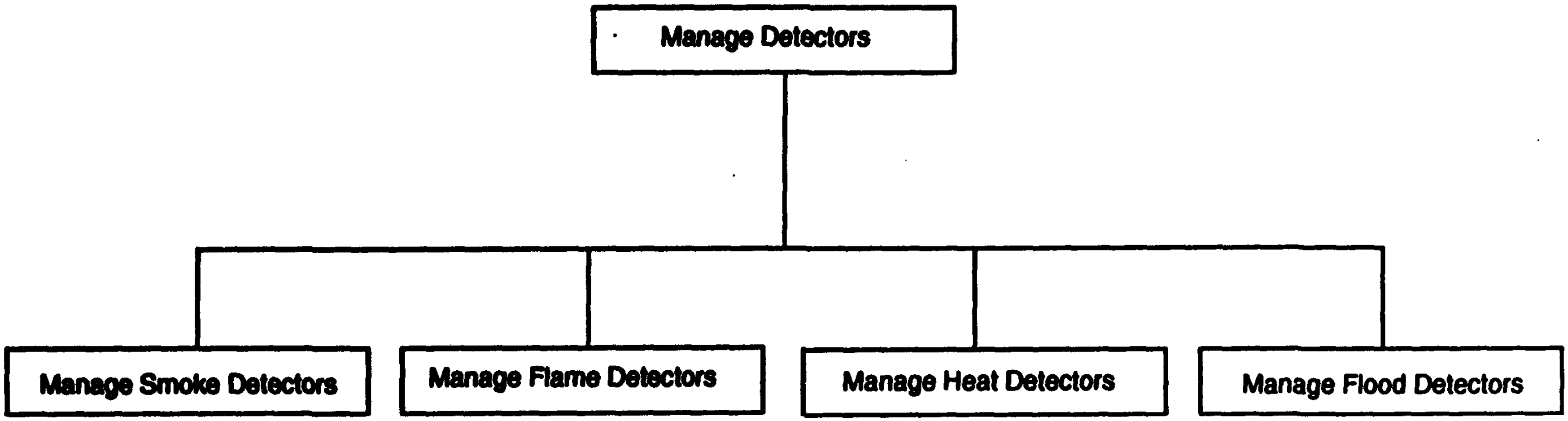


Level 2

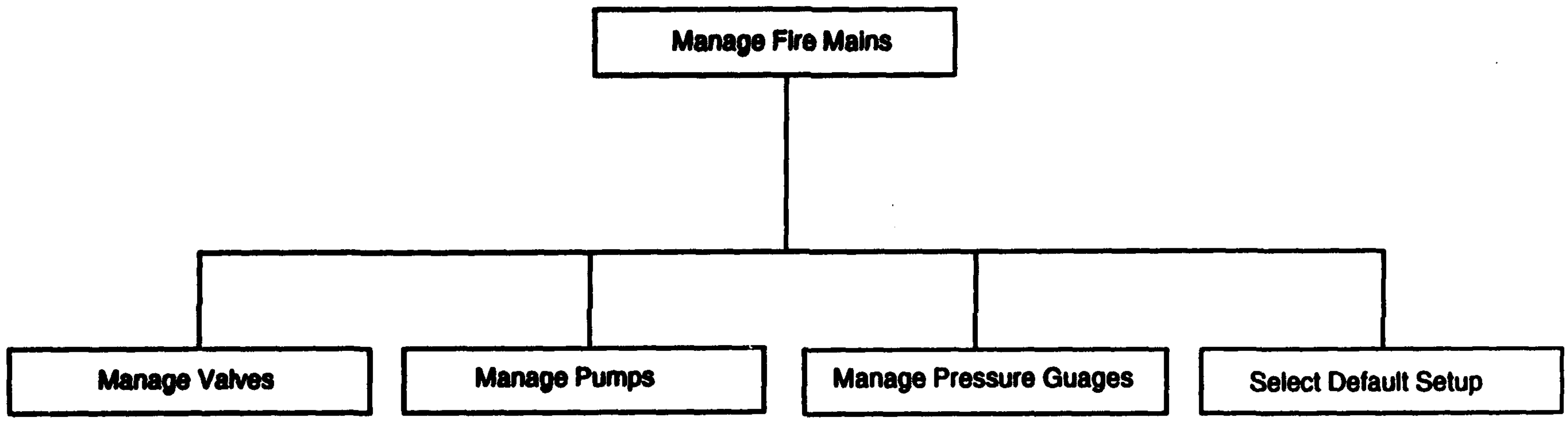


Set Time
Modify Hour
Modify Minutes
Modify Seconds
Modify AM/PM

Level 2

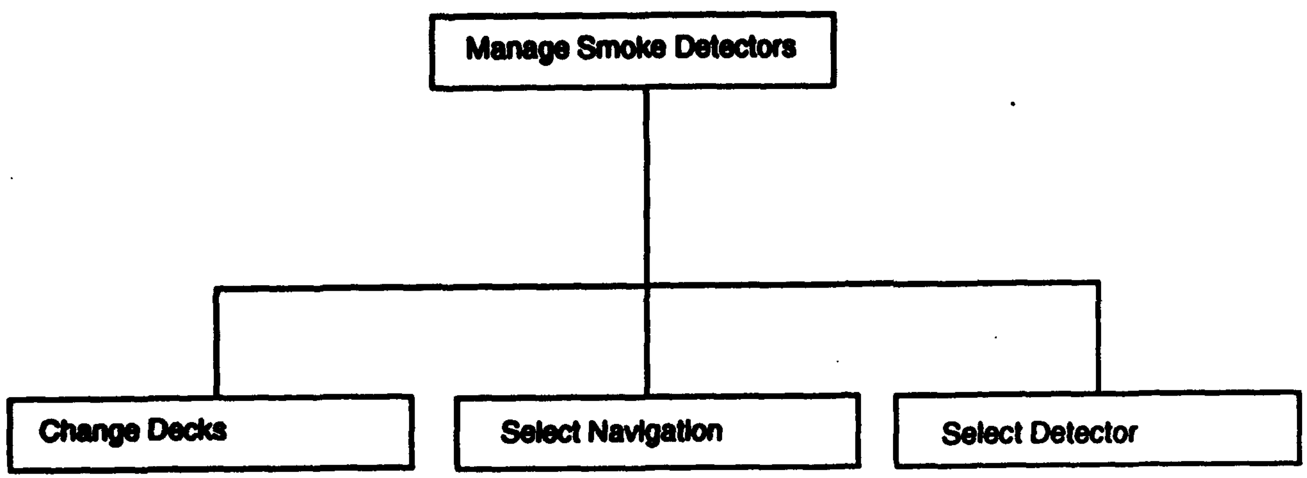

Manage Detectors
Manage Smoke Detectors
Manage Flame Detectors
Manage Heat Detectors
Manage Flood Detectors

Level 2



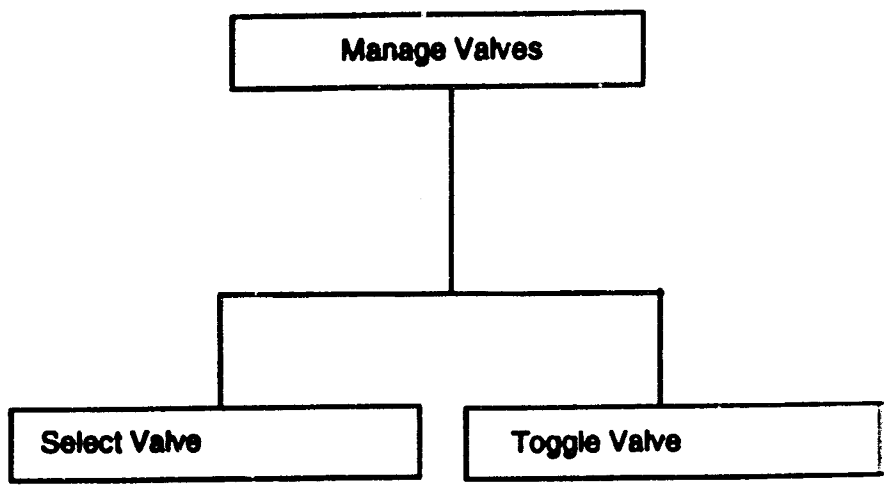
Manage Fire Mains
Manage Valves
Manage Pumps
Manage Pressure Guages
Select Default Setup

Level 3



Manage Smoke Detectors
Change Decks
Select Navigation
Select Detector

Level 3



Definitions:

Class: buttons
Description: objects appearing on the screen that look three dimensional
Highlighting: buttons appear two dimensional (!)

Group: toggle buttons
Description: bi-state button that changes state on selection
Highlighting: buttons show inverse video (!). Button text shows inverse state (!!).
Members: AM/PM

Group: default buttons
Description: rectangular buttons that control default configuration of fire mains
Highlighting: buttons become slightly darker on mouse down (!)
Members: Xray, Yoke, Zebra

Group: control buttons
Description: rectangular buttons that appear in control window(s)
Highlighting: buttons become slightly darker on mouse down (!)
Members: Local Time, Smoke Detectors, Flame Detectors, Heat Detectors, Flood Detectors, Fire Mains, Start, Stop, Xray, Yoke, Zebra

Group: status buttons
Description round buttons whose color describes the state of the object which they represent
Highlighting: buttons become transparent, i.e., show background color (!)
Members: Detectors, Valves

Group: pump buttons
Description round buttons whose color describes state of object they represent
Highlighting: these buttons take on a different color based on status of pump (!)
Members: Pumps

Group: alarm buttons
Description: square buttons that only appear when one or more of the detectors fire an alarm. Pressing and releasing the mouse button on these buttons has the same effect as pressing and releasing the mouse button on the status button for the given detector which fired the alarm. When in alarm state, the context of these buttons is the entire screen.
Highlighting: buttons are invisible unless an alarm has been fired in which case the button involved appears as a bright flashing red rectangle with the approximate location of the alarm shown on the ship display (!). These buttons show solid red (!!).

Group: deck buttons
Description: deck shaped buttons that represent various decks on a ship
Highlighting: buttons are normally gray outlined in blue but can turn dark blue (!) and blue and white striped (!!).

Group: valve and pump control buttons
Description: open/close, start/stop buttons that appear in mutually exclusive sets of two, one of which is always highlighted.
Highlighting: slightly darker gray (!), mutually exclusive

Group: gauge alarm status buttons
Description: square buttons whose color represents status of object they represent
Highlighting: buttons change color and text to match status (!).

Group: auto start buttons
Description: square check box that appears in pump control window
Highlighting: check appears in box (!).

Class: Unique buttons
Description: buttons which do not look three dimensional

Group: valve and pump status buttons
Description: square buttons whose color represents status of object they represent
Highlighting: buttons change color and text to match status (!).

Group: navigational buttons
Description: typical Macintosh buttons such as done, cancel, open, etc.
Highlighting: buttons show inverse video (!). If the mouse button is depressed and then dragged off the button icon, the icon will revert to normal and then revert to inverse if the mouse is brought back into the context of the icon.
Members: Done, Cancel, Undo

Group: increment/decrement buttons
Description: typical Macintosh button with an arrow on top and bottom
Highlighting: selected numeric item increments or decrements depending on location of the mouse and button shows inverse video(!).

Group: time buttons
Description: represent hours, minutes, and seconds with text in rectangular button
Highlighting: rectangular area turns brighter yellow (!). Text within rectangle shows effect of incrementing or decrementing (!!)

Task: Manage Damage Control				
Arena: Control Window				
User Action	Non-Mainline Action	Feedback/Display	Interface State	Notes
INTERLEAVED/ INCLUSIVEOR (Set Time				
Manage Detectors				
Manage Fire Mains				
Respond to Alarm)*				

Task: Set Time				
Arena: Time Window				
User Action	Non-Mainline Action	Feedback/Display	Interface State	Notes
Select Navigation(Time)		display time window all other navigation icons -!		
INCLUSIVEOR (Modify Hour				
Modify Minutes				
Modify Seconds				
Modify AM/PM)*				
	Select Button(Cancel)	erase time window	time not updated	ret. to Manage Damage Cont
Select Button(Done)		erase time window	time updated	

Task: Modify AM/PM				
Arena: Time Window				
User Action	Non-Mainline Action	Feedback/Display	Interface State	Notes
~[AM/PM button icon]				
Mv		am/pm button icon ! am/pm button icon !!	time period = opposite of previous	
M^		am/pm button icon-!		

Task: Modify Hour				
Arena: Time Window				
User Action	Non-Mainline Action	Feedback/Display	Interface State	Notes
~[Hour Icon]				
Mv		Hour Icon !		
M^				
~[increment/decrement icon]				
Mv		increment/decrement icon ! Hour Icon !!		
M^		increment/decrement icon -! Hour Icon -!!	Hour is changed	

Task: Modify Minutes				
Arena: Time Window				
User Action	Non-Mainline Action	Feedback/Display	Interface State	Notes
~[Minutes Icon]				
Mv		Minutes Icon !		
M^				
~[increment/decrement icon]				
Mv		increment/decrement icon ! Minutes Icon !!		
M^		increment/decrement icon -! Minutes Icon -!!	Minute is changed	

Task: Modify Seconds				
Arena: Time Window				
User Action	**Non-Mainline Action**	**Feedback/Display**	**Interface State**	**Notes**
~[Seconds Icon]				
Mv				
M^		Seconds Icon !		
~[increment/decrement icon]				
Mv		increment/decrement icon ! Seconds Icon !!		
M^		increment/decrement icon -! Seconds Icon -!!	Second is changed	

Task: Manage Detectors				
Arena: Control Window				
User Action	**Non-Mainline Action**	**Feedback/Display**	**Interface State**	**Notes**
INCLUSIVEOR (Manage Smoke Detector				
Manage Flame Detectors				
Manage Heat Detectors				
Manage Flood Detectors)*				

Task: Manage Smoke Detectors				
Arena: Main Screen				
User Action	Non-Mainline Action	Feedback/Display	Interface State	Notes
INTERLEAVED/ INCLUSIVEOR (Change Decks,				
(view not equal smoke detect: (Select Navigation (Smoke Detector))		Display (Smoke Detect. Figure) all other navigation icons -!		
Select Detector (Detector)		Display (Status Screen) detector icon-!		
Mv				these steps may overlap with
M^ not in [status screen]		Erase (Status Screen)		the next select detector task

Task: Manage Flame Detectors				
Arena: Main Screen				
User Action	Non-Mainline Action	Feedback/Display	Interface State	Notes
INTERLEAVED/ INCLUSIVEOR (Change Decks,				
(view not equal flame detect: (Select Navigation (Flame Detector))		Display (Flame Detect. Figure) all other navigation icons -!		
Select Detector (Detector)		Display (Status Screen) detector icon-!		
Mv				these steps may overlap with
M^ not in [status screen]		Erase (Status Screen)		the next select detector task

Task: Manage Heat Detectors				
Arena: Main Screen				
User Action	Non-Mainline Action	Feedback/Display	Interface State	Notes
INTERLEAVED/ INCLUSIVEOR {Change Decks,				
(view not equal heat detect: (Select Navigation (Heat Detector))		Display (Heat Detect. Figure) all other navigation icons -!		
Select Detector (Detector)		Display (Status Screen) detector icon-!		
Mv				these steps may overlap with
M^ not in [status screen]		Erase (Status Screen)		the next select detector task

Task: Manage Flood Detectors				
Arena: Main Screen				
User Action	Non-Mainline Action	Feedback/Display	Interface State	Notes
INTERLEAVE/ INCLUSIVEOR {Change Decks,				
(view not equal flood detect: (Select Navigation (Flood Detector))		Display (Flood Detect. Figure) all other navigation icons -!		
Select Detector (Detector)		Display (Status Screen) detector icon-!		
Mv				these steps may overlap with
M^ not in [status screen]		Erase (Status Screen)		the next select detector task

Task: Change Decks				
Arena: Main Screen				
User Action	**Non-Mainline Action**	**Feedback/Display**	**Interface State**	**Notes**
~[deck icon]				
Mv		deck icon !		
M^		(t=n)display selected deck deck icon -! deck icon !! all other deck icons -!!		

Task: Respond to Alarm				
Arena: Main Screen				
User Action	**Non-Mainline Action**	**Feedback/Display**	**Interface State**	**Notes**
		CONC(audible alarm, alarm button icon!)	alarm button is in alarm status	
Mv		alarm button icon-! alarm button icon!i	alarm button alarm status canceled	
M^		*display(status screen)*		
Mv				
M^ not in [status screen]		Erase (Status Screen)		

Task: Manage Fire Mains				
Arena: Control Window/Main Screen				
User Action	Non-Mainline Action	Feedback/Display	Interface State	Notes
view not equal Fire Mains: Select Navigation (Fire Mains)		Display Fire Mains schematic all other navigation icons -!		
INCLUSIVEOR (Manage Valves,				
Manage Pumps,				
Manage Pressure Gauges,				
Select Default Config.)*		Redisplay Fire Mains Schematic all other default buttons -!		

Task: Select Default Config.				
Arena: Default Selection Window				
User Action	Non-Mainline Action	Feedback/Display	Interface State	Notes
~[default button icon]Mv		default button icon !	selected = default button	
M^				

Task: Manage Valves				
Arena: Fire Mains View of Main Screen				
User Action	Non-Mainline Action	Feedback/Display	Interface State	Notes
Select Valve(valve)		(t=n)display valve window valve icon -!		
(Toggle Valve)				
~(x,y) not in valve window Mv				
M^		erase valve window	selected = none	

Task: Select Valve (valve)				
Arena: multiple				
User Action	Non-Mainline Action	Feedback/Display	Interface State	Notes
~[valve icon]Mv		valve icon!	selected = valve	the first two actions of this
				task may overlap the last two
	~[x,y] not in valve icon	valve icon-!		actions of the related manage
	~[valve icon]	valve icon!		valves task
M^			activated = valve	
	~[x,y] not in valve icon	valve icon-!		
	M^		activated = none selected = none	

Task: Manage Pumps				
Arena: Fire Mains View of Main Screen				
User Action	Non-Mainline Action	Feedback/Display	Interface State	Notes
Select Pump(pump)		(t=n)display pump window pump icon -!		
(Toggle Pump)				
~(x,y) not in pump window Mv				
M^		erase pump window	selected = none	

Task: Select Pump (pump)				
Arena: multiple				
User Action	Non-Mainline Action	Feedback/Display	Interface State	Notes
~[pump icon]Mv		pump icon!	selected = pump activated = pump	the first two actions of this
M^				

Task: Manage Pressure gauges

Arena: Fire Mains View of Main Screen

User Action	Non-Mainline Action	Feedback/Display	Interface State	Notes
Select Pressure gauge (pressure gauge)		(t=n)display pressure gauge window pressure gauge icon -!		
Operate Pressure gauge				
~(x,y) not in pressure gauge window Mv				
M^		erase pressure gauge window	selected = none	

Task: Operate Pressure gauge

Arena: Pressure Gauge window

User Action	Non-Mainline Action	Feedback/Display	Interface State	Notes
(OR(control alarm,				
control alarm delay,				
control pressure alarm,				
update alarm table)*				
select button(done)		erase window	update gauge information	
	select button(undo changes))*	undo changes	undo changes	

Task: update alarm table

Arena: Pressure Gauge window

User Action	Non-Mainline Action	Feedback/Display	Interface State	Notes
(~[check box icon]Mv		check box icon!		
M^)*				
~[update button icon]Mv		update button icon! (t=n)update button icon-!	table values updated	
M^				

Task: control alarm delay				
Arena: Gauge control window				
User Action	Non-Mainline Action	Feedback/Display	Interface State	Notes
OR(modify minutes,				
modify seconds)*				

Task: Control Alarm				
Arena: Gauge control window				
User Action	Non-Mainline Action	Feedback/Display	Interface State	Notes
(~[gauge alarm status button icon]Mv		gauge alarm status button icon!		
M^)				

Task: control pressure alarm				
Arena: Pressure gauge control window				
User Action	Non-Mainline Action	Feedback/Display	Interface State	Notes
~[slide handle icon]Mv				
~(x,y) in [slide]		slide handle icon > ~ in y axis color >> ~ within slide		
	~(x,y) not in [slide]	redisplay slide handle icon and color at previous location		
M^		redisplay slide handle icon and color at new location	level = new level	

Task: Select Pressure gauge (pressure gauge)				
Arena: multiple				
User Action	Non-Mainline Action	Feedback/Display	Interface State	Notes
~[pressure gauge icon]Mv		pressure gauge icon!	selected = pressure gauge activated = pressure gauge	
M^				

Task: Select Detector(button)				
Arena: multiple				
User Action	Non-Mainline Action	Feedback/Display	Interface State	Notes
~[button icon]Mv		button icon!	selected = button	the first two actions of this

				task may overlap the last two
	~[x,y] not in button icon	button icon-!		actions of the related manage
	~[button icon]	button icon!		detectors task
M^		button icon-!	activated = button	
	~[x,y] not in button icon	button icon-!		
	M^		activated = none selected = none	

Task: Select Button(button)				
Arena: multiple				
User Action	Non-Mainline Action	Feedback/Display	Interface State	Notes
~[button icon]Mv		button icon!	selected = button	
	~[x,y] not in button icon	button icon-!		
	~[button icon]	button icon!		
M^		button icon-!	activated = button	
	~[x,y] not in button icon	button icon-!		
	M^		activated = none selected = none	

Task: Select Navigation(button)				
Arena: multiple				
User Action	Non-Mainline Action	Feedback/Display	Interface State	Notes
~[button icon]Mv		button icon!	selected = button	
M^			activated = button	
	~[x,y] not in button icon	button icon-!	selected = previous button	
	M^		nothing activated	

Task: Toggle Valve				
Arena: Valve control window				
User Action	Non-Mainline Action	Feedback/Display	Interface State	Notes
OR((~[valve status button icon]Mv		valve status button icon! valve control buttons!		
M^),				
(~[valve control button-!]				
Mv^))*		valve status button icon! valve control buttons!		

Task: Toggle Pump				
Arena: Valve control window				
User Action	Non-Mainline Action	Feedback/Display	Interface State	Notes
OR(OR((~[pump status button icon]Mv		pump status button icon! pump control buttons!		
M^),				
(~[pump control button-!]				
Mv^)),		pump status button icon! pump control buttons!		
(~[auto start icon]Mv		auto start icon!		
M^))*		pump status button icon!		

APPENDIX 2

Most Frequently Used UAN Symbols

UAN SYMBOLS FOR THE USER ACTIONS COLUMN.

What is Represented	UAN Symbols	Meaning
Cursor movement	~	move the cursor
Object context	[X]	the context of object X, the "handle" by which X is manipulated
Cursor movement	~[X]	move cursor into context of object X
Cursor movement	~[x,y]	move cursor to (arbitrary) point x,y
Cursor movement	~[x,y]*	move cursor to zero or more (arbitrary) points x,y
Cursor movement	~[x',y']	move cursor to specific point x',y'
Cursor movement	~[x,y in A]	move cursor to (arbitrary) point within object A
Cursor movement	~[X in Y]	move to object X within object Y
Cursor movement	[X]~	move cursor out of context of object X
Switch operation	v	depress
Switch operation	^	release
Switch operation	Xv	depress button, key, or switch X
Switch operation	X ^	release button, key, or switch X
Switch operation	Xv^	idiom for clicking button, key, or switch X
String value	K"abc"	enter literal string, abc, via device K
String value	K(xyz)	enter value for variable xyz via device K
Grouping	( )	grouping mechanism
Sequence	A B	tasks A and B are performed in order left to right, or top to bottom
Repetition	A*	task A is performed zero or more times
Repetition	A+	task A is performed one or more times
Repetition	A^n	task A is performed exactly n times
Optionality	{A}	enclosed task is optional (task A is performed zero or one time)
Choice	| , OR	choice of tasks (used to show alternative ways to perform a task)
Repeating choice	(A | B)*	choice of A or B is performed to completion, followed by another choice of A or B, etc.
Order independence	A & B	tasks A and B are order independent (order of their performance is immaterial)
Interruptibility	A → B	task A can interrupt task B
Uninterruptibility	<A>	task A cannot be interrupted
Interleavability	A ⇔ B	performance of tasks A and B can be interleaved in time
Concurrency	A || B	task A and task B can be performed simultaneously
Waiting	A (t > n) B	task B is performed after a delay of more than n units of time following task A

UAN SYMBOLS FOR THE INTERFACE FEEDBACK COLUMN.

What is Represented	UAN Symbols	Meaning
Highlight	!	highlight object
Unhighlight	-!	unhighlight object
Highlight	!!	same as !, but use a different highlight
Location	@x',y'	at point x',y' (e.g., to display X)
Location	@X	at object X
Location	@x',y' in X	at point x',y' in object X
Display	display(X)	display object X
Erase	erase(X)	erase object X
Redisplay	redisplay(X)	erase X and display X again (in new location)
Outline	outline(X)	outline of object X
Dragging	X > ~	object X follows (is dragged by) cursor
Rubberbanding	X >> ~	object X is rubber-banded as its follows cursor
For all	∀	for all (e.g., ∀icons)