TABLE 7.2. 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

ACTIVITIES: Write a UAN task description for `manage calendar`, the task at the highest level of abstraction in the Calendar Management System. This task description should allow users to interleave major subtasks.

DELIVERABLES: One UAN task description for `manage calendar`, with appropriate temporal relations.

TABLE 7.3. 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)

CAUTION: Many people, when beginning task descriptions, get involved in low-level details. The high-level descriptions are actually easier and are very important; they often form the bulk of your UAN design representation, and they bring task analysis and design together. Don't overdo on this exercise. This step takes lots of time for a real system, and it overlaps with task analysis and design, too.

Possible Exercise Solution

Because we wish, in our design, to let a user interleave tasks at the highest level, the UAN task description begins something like this, which is a group of previously shown interleaved tasks:

TASK: **manage calendar**
(access appointment
⇔ add appointment
⇔ update appointment
⇔ delete appointment
⇔ establish alarm)$^{+}$

Interleaving (⇔) and repetition (+) are the primary temporal relations in this task description. This means that the user can perform the set of tasks that appear within the parentheses one or more times. The expression within the parentheses is represented at runtime by a set of interleaved instances of the `access appointment`, `add appointment`, `update appointment`, `delete appointment`, and `establish alarm` tasks.

Exercise—Developing More Detail in Design Representations Using UAN

GOAL: To use the UAN to represent some middle-level user tasks.

MINIMUM TIME: About 10 minutes.

ACTIVITIES: Think about how you might successively decompose some of the subtasks from the previous exercise. Write a UAN task description for `access appointment`. As already shown, this involves tasks such as searching and accessing various view levels of the calendar. Because these subtasks really aren't interleavable, you will need a different temporal relation.

DELIVERABLES: One UAN task description for `access appointment`, with appropriate temporal relations.

Possible Exercise Solution

The subtask breakdown of the `access appointment` task might be described in the UAN as the group of disjoint subtasks shown here:

TASK: **access appointment**	
USER ACTIONS	**INTERFACE STATE**
(search	
| access month	
| access week	
| access day)*	
access time slot	view level = time slot

As stated previously, it doesn't make much sense to think of interleaving `access month`, `access week`, and `access day`. For example, if a user is in the month view level, any attempt at user actions involved in the `access week` task would change the current view level to the week level, and so the `access month` task would no longer be active. So the repeating choice is more realistic and accurate in describing the temporal relation among these `access appointment` subtasks. To access an appointment, then, this task description specifies that the user does any number of `search`, `access month`, `access week`, and `access day` tasks, followed sequentially by a single `access time slot` task (time slots can be considered as containers of appointments).

Exercise—Further Detail in Design Representations Using UAN

GOAL: To use the UAN to represent another middle-level user task.

MINIMUM TIME: About 15 minutes.

ACTIVITIES: Write a UAN task description for the `access month` task. This involves establishing the month level as the current view level and navigating back and forth among months. To establish the month view level, a user must select some (any) month from the screen. Use a parameterized invocation of a general `select` macro task to represent user selection of a month in the calendar.

DELIVERABLES: UAN task descriptions for `access month` and for `select(object)`, with appropriate temporal relations.

Possible Exercise Solution

The `access month` task description should look something like the following:

TASK: **access month**	
USER ACTIONS	**INTERFACE STATE**
(select(any month)	view level = month
| move forward by month	
| move backward by month)$^+$	

The first subtask, `select(any month)`, allows a user to make the month view the current view level. By selecting a month on the screen, a user has instantiated the `select(object)` task, defined next. In this instantiation, the month object conceptually substituted for the `object` parameter, using the following task description for `select`:

TASK: **select(object)**		
USER ACTIONS	**INTERFACE FEEDBACK**	**INTERFACE STATE**
~[object icon'-!] Mv	object icon'!	selected = object
M^		

Exercise—More Practice with UAN

GOAL: To represent a couple more middle-level tasks.

MINIMUM TIME: About 10 minutes.

ACTIVITIES: Write a UAN task description for `access time slot`. This is similar to the `access month` task except that an initial condition of viability is added to ensure that the view is at the day level (you may remember this from Chapter 6). The user can then scroll up or down, finally selecting a time slot.

Next, write a task description for the `add appointment` task. This one is very brief.

HINT: Start by accessing an appointment, then editing it.

DELIVERABLES: UAN task descriptions for `access time slot` and `add appointment`.

Possible Exercise Solution

The `access time slot` task is a continuation of our decomposition of the `access appointment` task. It might look something like this:

TASK: access time slot	
USER ACTIONS	**INTERFACE STATE**
`view level = day:`	
`  ((scroll up | scroll down)*`	
`  select(any time slot))`	`view level = time slot`

As you may recall from the discussion in Chapter 6, this task begins with a *condition of viability* requiring the view level for navigation to be at the day level. The current view level value is kept as interface state information. Each task that accesses a view level will reset the current value of the view level state, as shown in the preceding task description.

Next is the second task of the highest-level task, `manage calendar`—namely, the task for adding a new appointment. Its UAN description could be as simple as:

TASK: **add appointment**
`access appointment`
`edit appointment`

Notice how you can build on the `access appointment` task here, applying abstraction, as the details of task decomposition slowly unfold.

We will skip the two tasks of `update appointment` and `delete appointment`, since they are very similar to the ones we've just done. Now look at the last of the five tasks in the `manage calendar` task.

Exercise—Connecting to the Articulatory Level of the UAN

GOAL: To represent near-articulatory level tasks.

MINIMUM TIME: About 20 minutes.

ACTIVITIES: Write a UAN task description for the `establish alarm` task, to notify a user when an appointment is impending. A user must initially be at the time slot view level. The first part of establishing an alarm is to set the alarm, associating it with a specific time slot or appointment. This can be followed with a subtask for setting the alarm parameters (e.g., lead time for notification).

You are also to write a UAN description at the articulatory level for the `set alarm` task. A user accomplishes this association of an alarm with a time slot by dragging a copy of the alarm icon (from the top of the screen in Figure 5.3) to the desired time slot.

DELIVERABLES: UAN task descriptions for `establish alarm` and for `set alarm`.

Possible Exercise Solution

The `establish alarm` task description might look something like this:

TASK: **establish alarm**
`view level = time slot:`
`(set alarm`
`set alarm parameters)`

As pointed out in Chapter 6, the condition of viability in the first line ensures that there is a current appointment (or at least a specific time slot) with which to associate the alarm. The second line is a macro that establishes the association of an alarm with the appointment in that time slot. The third line is the macro for setting the alarm parameters, such as alarm lead time (how far in advance of an appointment to sound the alarm), perhaps by means of a dialogue box.

In the Calendar Management System design, the `set alarm` task is accomplished by dragging a copy of the alarm icon from the upper left-hand corner of the screen to the time slot of the appointment. You might want to look back at the screen design in Figure 5.3 and see the alarm clock icon. This task can be written in UAN as:

TASK: **set alarm**		
USER ACTIONS	**INTERFACE FEEDBACK**	**INTERFACE STATE**
`view level = time slot:` `(~[alarm icon]` `Mv`	`alarm icon-!: alarm icon!`	`set alarm mode = on`
`~[time slot']`	`outline(copy(alarm icon)) > ~` `time slot'!`	
`M^)`	`@x',y' in time slot'` `display(copy(alarm icon))`	`set alarm mode = off`

The first line here contains a condition of viability for the whole task. The feedback for the action of releasing the mouse button (`M^`) in the last block indicates that a copy of the alarm icon is affixed to the appointment display at a specific point *x'*,*y'* (e.g., in the upper left corner) relative to the appointment itself.

This is the last task description we will develop for the Calendar Management System. The tasks already completed offer a start on the quasi-hierarchical task structure, as shown in Figure 7.1.

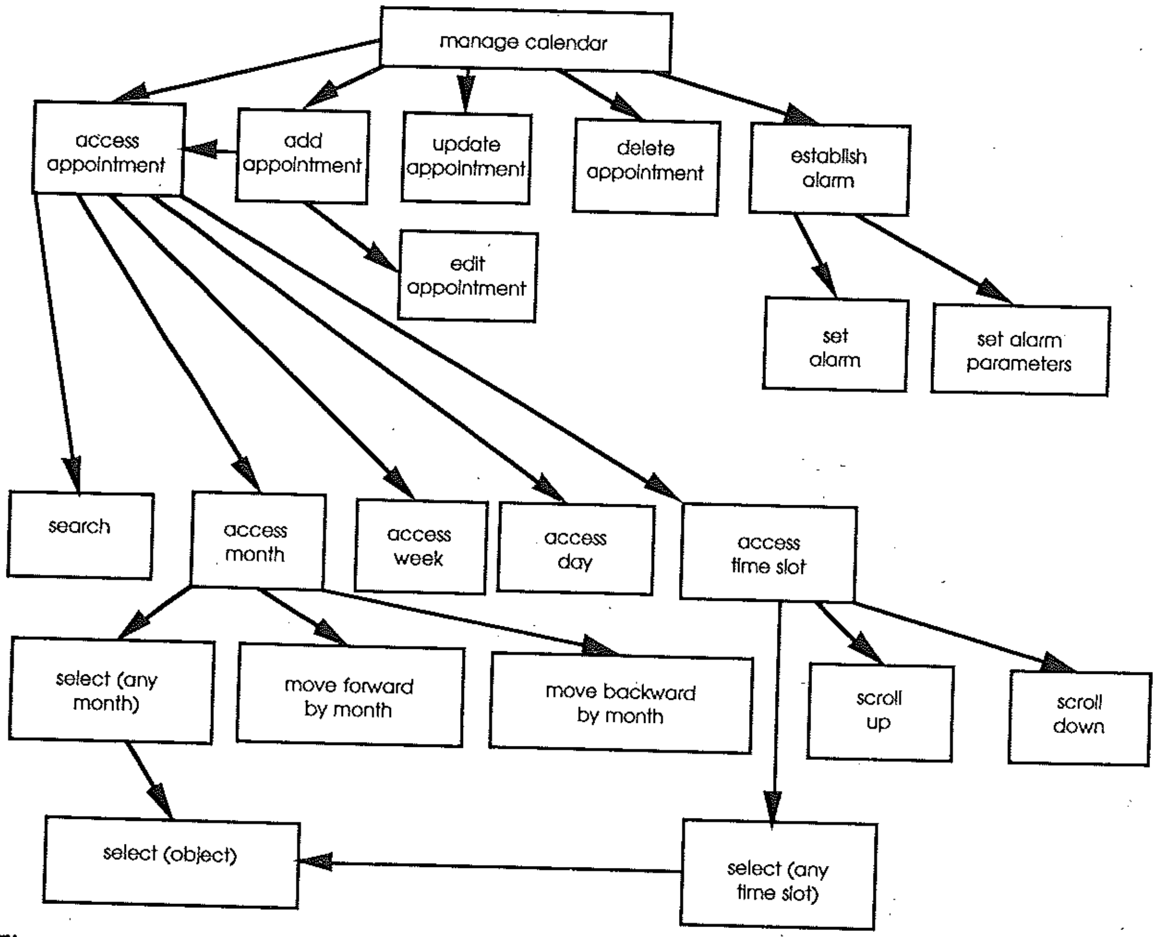

Figure 7.1. Beginnings of a quasi-hierarchical task structure for the Calendar Management System.

7.14 SUPPLEMENTARY REPRESENTATION TECHNIQUES

The UAN is effective in precisely and concisely describing user tasks. This is an important part of interaction design representation, but it is not complete. In addition, it is useful to complement UAN task descriptions with screen pictures and scenarios, interface state transition diagrams, and design rationale descriptions. Let's look at how we might use each of these other representation techniques in the Calendar Management System design.

7.14.1 Screen Pictures and Scenarios

The UAN is intended for describing user actions in the context of interaction objects, along with feedback and state information. Because the UAN does not show the appearance of screens and screen objects, it needs to be supplemented with pictures. For example, wherever the Interface Feedback column contains a description of something that is displayed, it usually should have a note that says, "See Figure XYZ," and Figure XYZ should accompany the design document. That is, UAN task descriptions should be supplemented with a set of screen pictures, such as the ones shown for the scenario designs in Chapter 5.

The following example, from the end of Section 7.5, continues the example of the task to `select multiple files`. One way this task allows a user to select is by dragging out a selection box to intersect or enclose the desired file icons. Section 7.5 didn't show visually how that would be done by a user. This could be difficult for an interface implementer to understand, even if it is completely and correctly written in UAN. Let's continue that example by using a screen sketch to accompany the UAN task description and clarify the design. Consider the following description of a task for selecting multiple files by dragging out a selection box:

TASK: **drag box multiple select**		
USER ACTIONS	**INTERFACE FEEDBACK**	**INTERFACE STATE**
`~[x',y'] Mv`		`x',y' is fixed corner of rectangle`
`~[x',y']*`	`dotted rectangle >> ~` **`See figure "selection rectangle"`**	
`~[x'',y''] M^`	`objects intersected and enclosed by rectangle are !`	`selected = all intersected and enclosed objects`

Notice the note in the Interface Feedback column referring to the figure called "selection rectangle." This figure would be part of the scenario design and would appear somewhat like Figure 7.2.

Note that the screen picture of Figure 7.2, showing an example layout of screen objects surrounded by the drag box (the dotted rectangle), is also annotated with

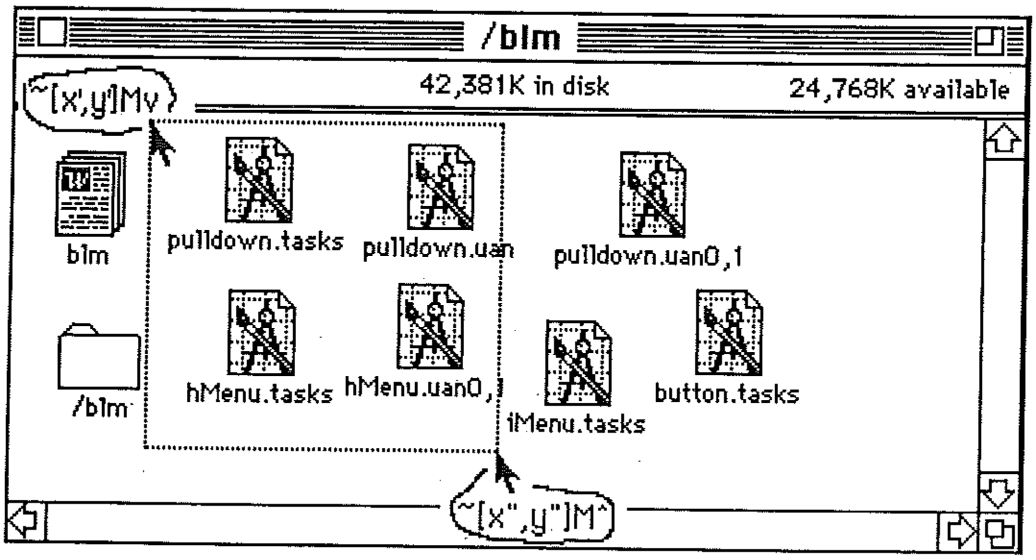

Figure 7.2. Scenario figure called "selection rectangle" From Hartson, Siochi, & Hix, *ACM Transactions on Information Systems*, p. 168. © 1990. Reprinted by permission.

UAN descriptions. Often UAN symbols can enhance the descriptive power of a screen picture by describing accompanying user actions.

7.14.2 Interface State Transition Diagrams

Many user interaction designs have lots of interface states. Often, the states are viewed by a user as modes. While the UAN has a column for Interface State, this often is not enough to represent a clear global picture of states and state transitions in an interaction design. Section 4.3 stated that early design activities can involve producing a state transition diagram to indicate sequencing and state or mode information. Indeed, it can be useful to supplement UAN task descriptions with *interface state transition diagrams.*

The different views within the Calendar Management System are modes. The way a user gets back and forth among them is by user actions that cause state transitions. The relationships among the views, and their corresponding modes, are not necessarily easy to see from individual task descriptions or scenario designs, so this provides an opportunity for us to improve the communication power of our design representations.

Exercise—Design Representation Using A State Transition Diagram

GOAL: To produce an interface state transition diagram to supplement the UAN task descriptions and scenario designs as a more complete set of design representations of the Calendar Management System.

MINIMUM TIME: About 30 minutes.

ACTIVITIES: Sketch a state transition diagram with the various views (which are states or modes of some kind in your design) as nodes, and with user actions that take a user to different modes or states as arcs.

DELIVERABLES: A state transition diagram drawn out on paper.

Possible Exercise Solution

Navigation among views within the Calendar Management System interaction design is shown in the diagram of Figure 7.3. There is clearly more than one possible correct solution, so your state transition diagram for the Calendar Management System design may differ from this one.

In this state transition diagram, the "START" arrow indicates that the month view is the default view; when the system is brought up, the month view is on top and therefore visible to a user. It is also easy to see in this diagram that, once in the month view, a user can go forward and backward among months. The same is true for the week and day views, going forward and backward among weeks and days, respectively. A user gets from either the month view or the day view to the week view by selecting any week. If some instance of a week is visible on the desktop, a user can click on it and bring it to the top, making the week view current. The month, week, and day views are identical with respect to all these navigational operations.

The day view has some additional features. This is the only view from which a user can move, by clicking on any time slot, to the time slot view, where appointments are kept. From here, a user can move to the type/edit state simply by typing on the keyboard. A user can return to the month, week, or day view from either the time slot view or the type/edit state by selecting (clicking on) any month, week, or day that is available from the desktop.

7.14.3 Design Rationale Descriptions

Because the design representation is a working document, as well as the means for communication among developer roles, designers would be wise to include, as part of the interaction design representation, their *design rationale*—comments about trade-offs they have faced and reasons behind their design decisions. These comments supplement UAN task descriptions, scenarios, and state diagrams, as a more complete representation of interaction designs. In an iterative development process involving several people in various roles, it is important to know *why* a certain design feature was used (and why others were rejected). If the feature itself fails in usability testing, the people in charge of redesign must understand the original goal of the feature. At a later time, the maintenance process also benefits from information on design decisions, often preventing repetition of an unsuccessful design previously tried and rejected by designers.

Implementers and evaluators also use design rationale descriptions to comment as early as possible on the design. Implementers are in the best position to estimate the cost of implementing certain features; evaluators must document their reasons

n interaction
ore than one
lendar Man-

t the month
ew is on top
, once in the
same is true
s and days,
o the week
ie desktop,
irrent. The
ivigational

m which a
e appoint-
by typing
om either
y month,

ieans for
lude, as
mments
s. These
ms, as a
opment
why a
re itself
nd the
enefits
iccess-

iment
imate
asons

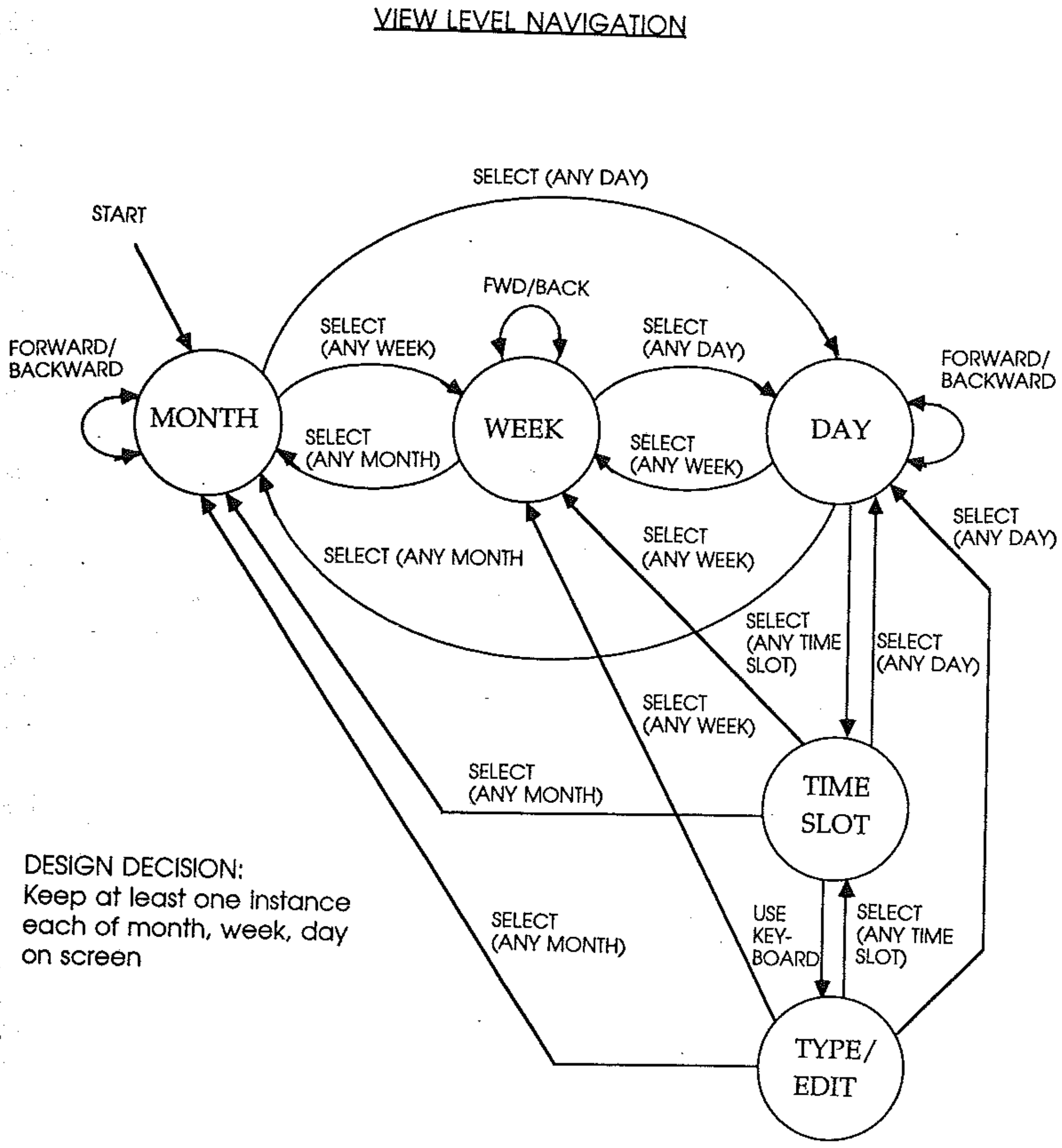

Figure 7.3. State transitions for navigating among views of the Calendar Management System. Adapted from Hartson & Gray, *Human-Computer Interaction*, pp. 1–45. © 1992. Reprinted by permission.

for suggested design changes. We do not offer any special notation for recording these decisions; use prose in whatever level of detail you deem necessary. Then, like screen sketches and figures, these descriptions can be referred to directly within any UAN task description to which they might be relevant.

7.15 FINAL UAN EXERCISE

Now it is time for you to write some UAN task descriptions for the Calendar Management System design you produced in Chapter 5. You probably didn't work out all the details of the design then. You will have to make some more design decisions now for those parts that are incomplete.

Exercise—Design Representation Using UAN

GOAL: To use the UAN to represent your own Calendar Management System design.

MINIMUM TIME: About one hour (obviously, this can take a lot longer; do as much as you like; once again, the goal is not complete solutions, but insight into and practice with the process).

ACTIVITIES: Produce several high-level and lower-level (articulatory) task descriptions using the UAN for your own scenarios and design that you produced in the exercise of Chapter 5. Also, try a state diagram for your own design.

DELIVERABLES: Several UAN task descriptions and a state transition diagram for your Calendar Management System design.

7.16 EXTENSIONS TO THE UAN

As already mentioned, the UAN has been left open intentionally, so that user interaction designers can invent their own UAN features to address their own particular task representation problems. In real-world use of UAN, almost all UAN users add a column for Comments, for prose explanations of complicated or potentially confusing steps in UAN descriptions.

Some users add a column for System Actions, making the relationship between system and user more symmetric in the interaction representation. This accommodates the need to represent actions that occur autonomously (not in response to a user action) within the system, but that have feedback in the interface to let a user know they have happened. For example, the system might need to announce the arrival of new mail or might simply need to update the display of a clock.

Some other extensions have also been tried. As the UAN has been presented here, user actions are limited to physical actions such as keystrokes and mouse movement. It could also be argued that the Interface Feedback column, which shows the system response to user actions, corresponds closely to desired perceptual actions of the user. To further support task analysis, columns sometimes have been added for user intentions and goals. Adding columns for cognitive actions, such as memory actions (e.g., recall, memory loading, closure) and decision making, have been done, to direct designs in supporting users' cognitive needs during task performance. This comes under the heading of ongoing research.

7.17 SOFTWARE TOOLS FOR UAN SUPPORT

It is obvious to most users of the UAN that its use could be greatly enhanced through software support tools. We (and others) are exploring a number of different tools to support the use of UAN in interaction design. The most fundamental UAN support tool is a UAN editor that developers can use to enter their interaction designs into the rows and columns of a UAN table. This aspect of the tool is similar to an engineering spreadsheet with simple text editing inside each cell and the ability to set the size of a row or column. Graphics editing facilitates development of screen pictures and scenarios. Finally, hypertext connections help a designer automate references from UAN descriptions to notes, figures, and state diagrams. The tool also supports some customization of UAN, such as the addition of new columns.

This UAN tool is part of a larger integrated set of tools to serve as a broad user interface development environment. This larger environment is called the Interface Development Environment and Analysis Lattice, or IDEAL; it will eventually have components to support much of the development work that occurs in the behavioral domain, including systems analysis, design and design representation, rapid prototyping, usability specifications, and formative evaluation. IDEAL is discussed in more detail in Chapter 10, on formative evaluation. Other tools presently being considered for UAN support include tools for analytic evaluation of interface usability, code generation and translation, and generation of end-user documentation from UAN task descriptions.

7.18 EXPERIENCE WITH THE UAN

As mentioned in Chapter 6, when we first began developing the UAN, we were doing it for our own internal need to communicate behavioral descriptions of interaction designs to implementers for construction and to evaluators for a preprototype view of the design. We used the UAN to conduct walk-throughs of the interaction design, and to check implementation against the design. We estimated that approximately 80% of the design of one system was implemented exactly as specified in UAN. Of the 20% that did not conform to the UAN design, 10% was due to misinterpretation of the UAN by the implementers, and 10% was due to their simply not wanting to implement it as represented. Other teams that have used the UAN report similar results.

To determine more about the usefulness of the UAN, we have promoted its use in commercial, government, and academic user interface development environments. The UAN has been used for the design of a telephone interface. A large government agency used the UAN to represent samples of a desired look and feel in a request for proposals (RFP) and to represent some prototype designs (Hix, Hartson, Siochi, & Ruppert, 1993). An interaction designer at a military installation, struggling through a several-inch-thick stack of human–computer interaction guidelines, used the UAN to document these guidelines precisely and to prevent repeated rereading of many lines of prose.

The UAN was also used to represent several multimedia interfaces involving full-motion video and audio for a digital video interactive (DVI) application being developed for a commercial company. Another multimedia system, called Project Envision, involves a database of computer science literature, including full text, audio and video clips, and animated algorithms. Interaction designs for a user interface management system and some interactive systems in Project DRUID at the University of Glasgow (Scotland) have been represented with the UAN. An interaction designer in Britain also used it to describe precisely what happens during a complex scrolling-in-a-window task for a new design. Designers at a commercial company used the UAN for the interface of a graphical editor. A data flow configuration system has been designed with the UAN at the Jet Propulsion Laboratory.

The UAN has been used by a producer of large-scale commercial security systems, such as might be used in airports and banks. The UAN was introduced into this project after a first design was complete (according to the designers). A number of significant design deficiencies were discovered while reverse engineering that design into UAN from the 400^+-page prose-and-pictures design document. This caused the development team to rethink task analysis and to do substantial redesign. The UAN was also used as a review and presentation technique for walk-throughs during development. One of the development team members translated UAN task descriptions directly into user documentation. This diversity of uses of the UAN indicates the broad range of interaction styles it can represent.

The UAN has proven to be effective in conveying large and complex user interaction designs from designers to implementers and evaluators. The UAN has been found by many of its users to be expressive and highly readable because of its *simplicity*, natural enough so that it is *easily read and written with very little training*. In a typical training session, participants are reading UAN within half an hour and writing simple UAN task descriptions within one hour. In places where hands-on training for UAN has not been possible, developers have been provided with a tutorial containing much less detail than in Chapter 6 and this chapter, and they have picked up an adequate amount of UAN to get started.

User interaction developers find UAN symbols and idioms to be *mnemonic* and *intuitive*. Perhaps more important, they like the *thoroughness* of the descriptions and feel that it *facilitates communication* among interaction designers and implementers. They find it to be *precise*, *concise*, and easily *extensible*, as needed for their particular environment.

Among the negative comments are concerns that UAN descriptions may be too detailed to be used early in the design process, when too much specificity may limit a designer's creativity. One or two developers have commented that the symbols should be more expressive (e.g., they wanted to change the ~ to MOVE TO), or that the symbols or columns did not allow them to fully represent their design. For this reason, you should feel free to extend and customize UAN symbols, columns, or any other feature, so that the UAN technique will more completely support your own individual user interaction design needs during representation.

7.19 SUMMARY

Now that you have worked your way through two chapters on the UAN, you should have a much better appreciation for why such a behavioral design representation technique is needed in the user interaction development process. Now you can see how the precision of the UAN can help you alleviate many of the problems of ambiguity and incompleteness that are inherent in other design representation techniques, particularly prose. Regardless of the number of people on your interface development team (even if it's just you), you need some sort of technique for capturing designs as a record of decisions and for handing them over to a programmer. Even if you do the programming, too, you need a record of what you designed when it is time to implement the user interface.

If you feel you do not need the detail offered by the UAN for your entire interaction designs, perhaps you will find it useful for representing at least some portions of your designs. Some serious UAN users use mostly the levels above the articulation level, leaving the articulation level to widget definitions. We welcome feedback on your use of the UAN, and any suggestions you might have for its extension.

REFERENCES

Buxton, W. (1983). Lexical and Pragmatic Considerations of Input Structures. *Computer Graphics, 17*(1), 31–37.

Hartson, H. R., & Gray, P. (1992). Temporal Aspects of Tasks in the User Action Notation. *Human-Computer Interaction, 7*, 1–45.

Hix, D., Hartson, H. R., Siochi, A. C., & Ruppert, D. (1993). The Customer's Responsibility for Ensuring Usability: Requirements on the User Interface Development Process. To appear in *Journal of Systems and Software.*

Thimbleby, H. (1990). *User Interface Design.* New York: ACM Press/Addison-Wesley.