•6• *The user's task*

Tasks have structure too

The first displays that we analysed at in this guide required users to operate 'control panels' for lighting and heating (Figure 1.8). The two designs differed in the way the on and off buttons were grouped: as columns of similar buttons, or in rows with the room label. Although we suggested that the 'rows' design was better for a user who had to control rooms individually, someone who had to check that the lights or heating were on or off in the whole building would benefit from the column grouping. This showed that it was necessary to consider the task that the user was performing, because this would affect the order in which they would want to move between the screen objects.

Figure 6.1 uses a structure diagram to show the difference between these two tasks. To 'check a room', the user has to find the room, and then check the buttons. To 'check the building', the user needs to successively check buttons, but doesn't need to look at the room numbers.

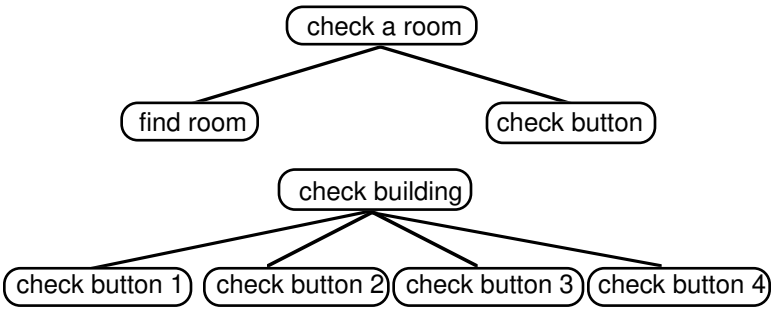

Figure 6.1: the structure of the tasks involved in using the 'control panels' of Figure 1.8

Having used the structure diagram to represent these tasks, we could also use the transition path diagram to show the sequence in which the steps of the task are carried out (Figure 6.2). These are particularly simple tasks, and as we have shown them, just require one transition 'down' into the structure of the task before the separate steps are carried out. By explicitly showing the order that those steps are carried out, they do give us the information that we need to choose between the 'row' and 'column' designs of the control panels.

In the first task, the user has to first find a room, and so the display must be structured to support a visual transition from the panel to the room numbers, followed by a transition from the room numbers to the related buttons. In the second task, the user has to locate the set of relevant buttons, and then must be able to make a sequence of transitions through them.

Until now we have emphasised that the structure diagrams and transition path diagrams represent the 'objects' that the user is perceiving on the display, and have linked this to the concept of a mental object representation. Clearly, the steps of a task are not objects, and yet we have shown that they can also be represented by the diagrams.

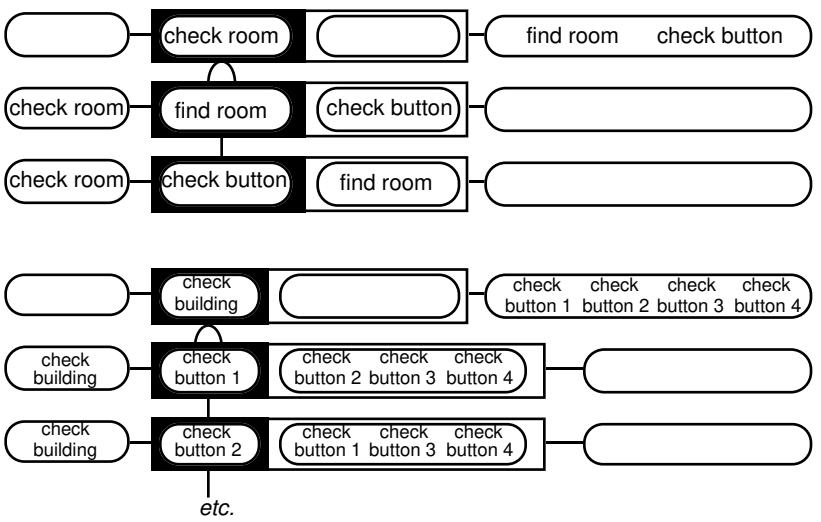

Figure 6.2: transition path diagrams for the tasks involved in using the 'control panels' of Figure 1.8

In our discussion of the icon search task, where people had to locate an abstract or representational icon from the arrays shown in Figure 5.1, we explained that a propositional representation of the 'target' icon was used to control the search: each icon was compared against it until a match was found. We went on to describe how propositional 'knowledge' could affect the perceived object structure, and showed how the use of two different button layouts in a hypertext (Figure 5.4) affected the propositional knowledge that people developed. As you might expect, task structures are also propositional representations. We can use the structure diagrams to represent the breakdown of tasks into steps (and task steps into smaller steps, if necessary) in the same way that objects can be broken down into smaller, component parts.

Task structures are propositional representations, and can be analysed using structure diagrams and transition path diagrams.

This is important in designing displays because of the feedback loop between propositional and object representations, as shown in Figure 2.2. Many of the examples we have shown in the last few sections have indicated that the propositional expectations about a visual figure can affect the object that is actually perceived. These examples were equivalent to single-step tasks, where the propositional representation was just some expected shape. In tasks that have several steps, the subject of the propositional representation will change to represent each step in turn. Just as transitions can be made through an object structure, zooming-in to objects and stepping through their component parts, transitions can also be made through a propositional structure, as the user concentrates on the steps of a task and tries to carry them out in the right order.

The design of the display can help the user make these transitions by providing object representations that match the structure of the task. When the user needs to choose one from several task steps, a display that has the appropriate screen element forming the psychological subject of the user's object representation will be providing helpful information through the object-to-propositional transformation.

An object representation that matches the task structure can help users form the appropriate propositional representation of the task

This is the reason why the 'lighting controls' of Figure 1.8 are better suited for the task of controlling individual rooms, while the 'heating controls' are better for controlling the whole building. The rows in the lighting controls support the transition from the room label to the relevant button, while the columns in the heating controls support the transition from button to button, as indicated in Figure 6.2. As the evidence from the use of the hypertext system showed, the way that users will come to think about a task can be influenced by the display structure. Designers of displays can take advantage of this to help users from propositional task structures that will be easy to recall, as the next few examples will show.

Ambiguity in task structures

Imagine someone who uses an ordering system in which they receive requests and have to process them, in a strict order (this system is one that has been used in experiments, but is based on real applications). The requests contain four pieces of information that need to be 'established' (e.g., copied into another form) and 'actioned' (e.g., sent to another department). The eight steps are:

1. **Display** the *time* the message was received
2. **Stamp** the order with this *time*
3. **Locate** the *originator's* address
4. **Confirm** receipt of the order to the *originator*
5. **Index** the *register* number of the ordered item
6. **Store** the *register* number on the order form
7. **Identify** the address of the *recipient*
8. **Dispatch** the order to the *recipient*

The bold word in these steps is a command name that the user has to type into the system to perform the operation. Figure 6.3 shows two ways that the user could do this task. In the first structure, the user establishes each piece of information and actions it immediately before going on to the next piece, as in the sequence given above. This produces four 'pairs' of task steps, which we have labelled with the piece of information that is being established and actioned in each pair (shown in italics in the sequence above). In the second structure, the user carries out four 'establish' operations to get all of the pieces of information from the message (steps 1, 3, 5 and 7 above), and then 'actions' them all (steps 2, 4, 6 and 8). Either of these sequences seems plausible, but the system must be designed so that only one is allowed: which one should be chosen?

To answer this, look at the two transition path diagrams that correspond to the use of these structures (Figures 6.4 and 6.5). These show that the 'two groups of four' structure requires thirteen transitions overall, while the 'four pairs' structure requires seventeen transitions. At first sight it might seem that carrying out all the establish operations first, and

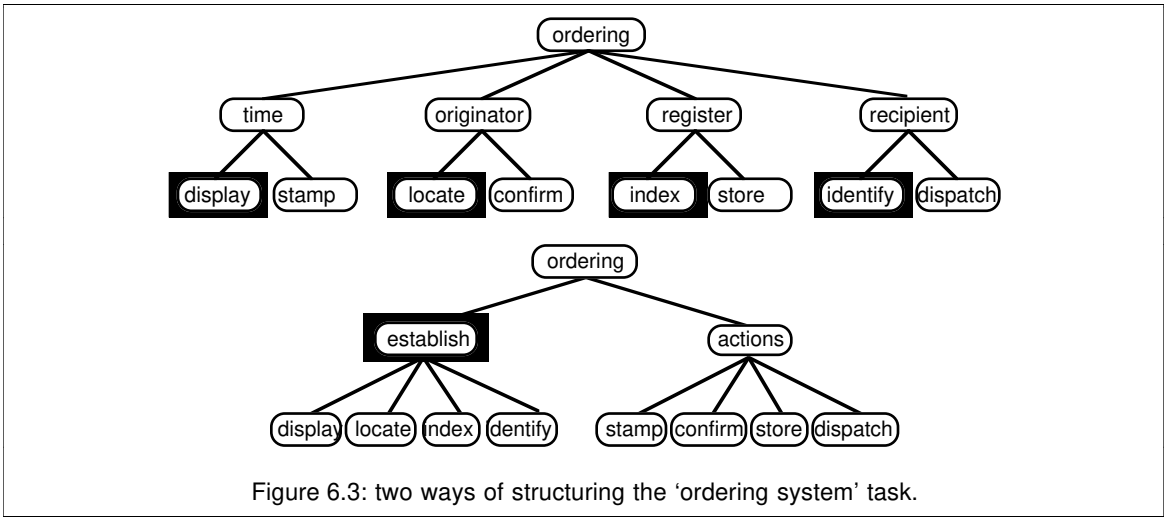

Figure 6.3: two ways of structuring the 'ordering system' task.

then the action operations, is the shortest, and so best solution, because the user doesn't have to make so many transitions up and down the structure. This is not the whole story, though: as with transitions through an object structure, we need to consider how 'difficult' each transition is.

In Figure 6.4, the first transition is from the task overall to the group of 'establish' operations. In Figure 6.3 we have shown this as the pragmatic subject, because you have to establish information before it can be actioned. This means that the transition will be made directly to that part of the task. In other words, it is unambiguous. The next four transitions must be made into and then between the four establish steps (display, locate, index, and identify). There is no pragmatic reason why any of these steps should be performed before any of the others: the user simply has to learn their order.

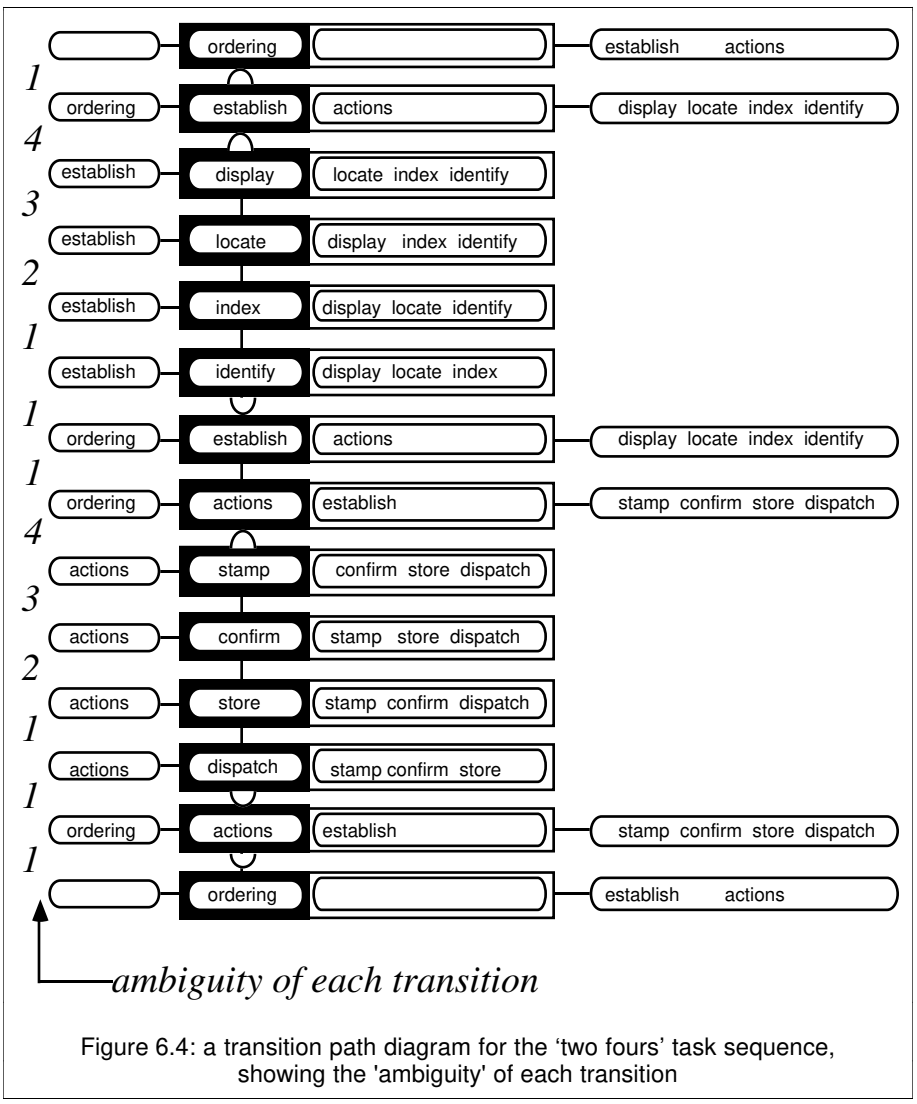

Figure 6.4: a transition path diagram for the 'two fours' task sequence, showing the 'ambiguity' of each transition

If the user cannot remember what order these four steps are in, they might make an error. Of course, you might expect them to remember one or more of the steps, but the more options there are, the greater the chance of their making an error by missing one out or choosing the wrong one. For the first transition, there are four steps to choose from, and so in Figure 6.4 we have indicated this by the number 4 to the left of the transition. This is a measure of the 'ambiguity' of the transition. If they get the first step right, there are then three options to choose from for the second step, and if they get this right,

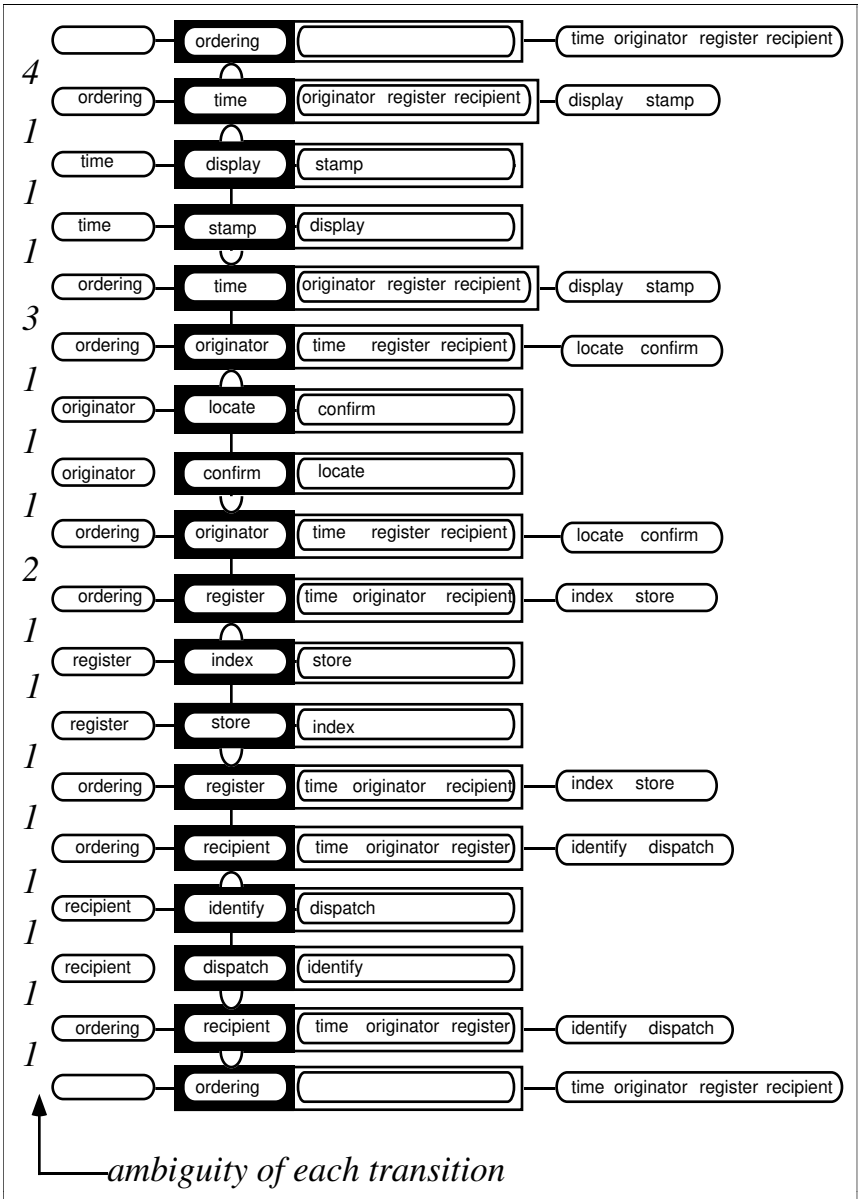

Figure 6.5: a transition path diagram for the 'four pairs' task sequence. showing the 'ambiguity' of each transition

there are then two left for the third step. Of course, if they get the first three steps correct, there is only one option to choose from for the fourth step, and so this step would be unambiguous (i.e., it has an ambiguity of 1).

The next two transitions are back 'up' to the establish group, and 'across' to the actions group. Both of these are unambiguous - in each case there is only one alternative to choose from. Then there are another four transitions through the action steps (stamp, confirm, store and dispatch). Again, there is no pragmatic reason for these to be carried out in the required order, and the 'ambiguity' of each of these transitions is 4, then 3, then 2 and then 1, as for the 'establish' steps. To calculate the 'chance' of a user getting all of these steps right simply by guessing at each transition, we have to multiply all of the ambiguity values together. For Figure 6.4, ignoring the 1s, this is 4 x 3 x 2 x 4 x 3 x 2, which makes 576. This means that if you tried to guess the task sequence for the 'two fours' condition, there are 576 different plausible sequences, only one of which is correct.

Looking at Figure 6.5, we can see that although the 'four pairs' structure results in the same eight steps being carried out, the 'ambiguous' transitions happen at different points in the path through the task sequence. Now the first transition, from the ordering task down to the first pairing, 'time', has an ambiguity of 4, since there are four pairs to choose from. The next two transitions, through its two steps, are unambiguous, because the 'action' (stamp) cannot be performed until its information has been 'established' (display). There is then a transition back up, which is unambiguous (the steps only belong to the 'time' group) and one across to the next pair, 'originator'. This transition has an ambiguity of 3, since there are three remaining pairs that could be chosen. Again, the transitions through the 'locate' and 'confirm' pair, and than back up to 'originator' are unambiguous. There is now a choice of two remaining pairs, and so the ambiguity of the transition to 'register' is 2, but all of the remaining transitions are unambiguous, including the transition to the final pair, 'recipient', since it is the last one left. This gives a total ambiguity of the path of 4x3x2, or 24, much less than the 'two fours' path.

The **ambiguity** of a transition is a rough guide to the difficulty users will have in making it correctly

These ambiguity measures are only rough guides to the difficulty that users will have in remembering a path through a sequence, but they do allow different options to be compared by emphasising the amount of support the user will require to 'disambiguate' their choices at each transition. In some cases, choices can be supported by the use of 'pragmatic' knowledge about options, such as the constraint that 'establish' steps must come before 'action' steps. In other cases, users can be supported by the structure of information on the display. If, instead of simply having to type the eight steps

in the ordering task, users could click on buttons, the ordering of the buttons on the display could be used to provide information about the required order of the commands.

In practice, you don't always need to construct the complete transition path diagrams to calculate a task structure's ambiguity. The numbers to be multiplied can be obtained directly from the structure diagrams. We drew the transition path diagrams here to show what the measure really meant in terms of the demands that the interface design was placing on the user's cognition – and they could equally well help in teasing apart difficulties in more complex situations, or in persuading other designers about a problem.

Conceptual structures

An important point to note about propositional representations is that the clustering of task steps into groups can be purely 'conceptual': the middle level in the structures of Figure 6.3 has no command or button-press associated with it. This level simply reflects the way that the users have been taught the task. It is part of their mental model of the task, not part of the real world. All the computer system cares about is receiving the eight commands in the right order.

Because people can learn the structure of tasks, it can be beneficial to give different tasks within a system a common structure. This can reduce ambiguity by removing the need for users to choose between different structures. Imagine a Cash Machine that, once the user's card has been inserted and their secret PIN number verified, allows people to either press a button to accept out a default amount of money (£50), or to press another button to request a specific amount (e.g., £30 or £60). This requires people to learn two task structures, depending on whether they want the default amount or a different amount, as shown in Figure 6.6.

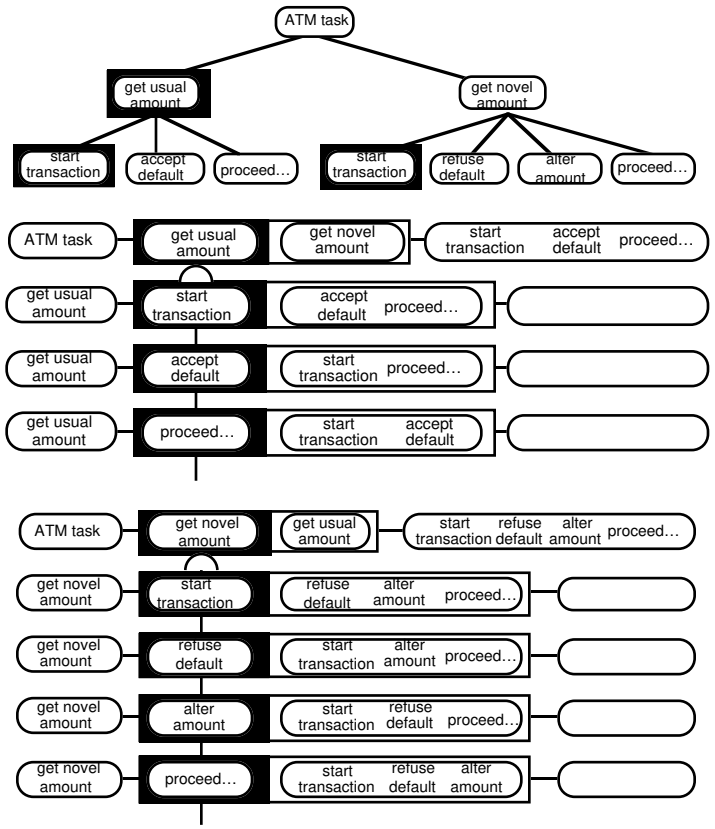

Figure 6.6: the task structures and transition path diagrams for accepting the default amount of cash or rejecting it and requesting a different amount.

There is no ambiguity in this task structure, since the order of each of the steps is constrained for pragmatic reasons. It differs from the 'ordering' system in that not all of the steps have to be carried out: only one of the two sub-tasks needs to be carried out each time the Cash Machine is used. This is fine for users who want to 'get usual amount', since as it is the most frequent action that perform with the machine, it will be the pragmatic subject. On the occasions that they don't want £50, though, they will have to remember to carry out a different set of steps, but one that is not immediately

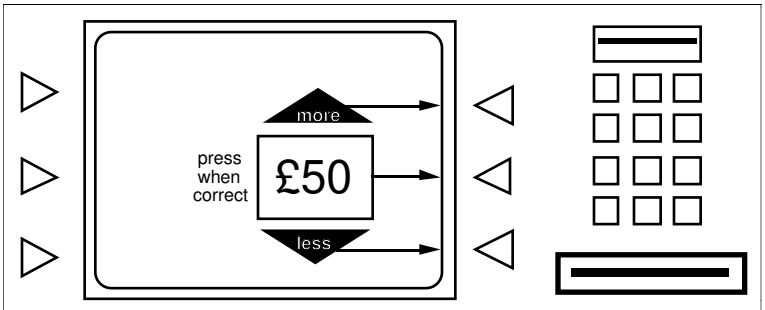

Figure 6.7: a redesigned Cash Machine with a single task structure

different. In fact, once they have gone through the 'start transaction' step, they may simply carry on with the rest of the most frequent sequence, and 'accept default', even though it isn't what they wanted.

Suppose the machine were redesigned, so that instead of pressing one button to accept the default, and another to go on to a different dialogue where they could specify a different amount, both tasks were merged into one. In the machine shown in Figure 6.7, the user can press the middle button to get the amount shown, or they can press the upper or lower buttons to increase or decrease the amount shown, and then press the middle button.

This combines the task structures for taking default and novel amounts into one structure, as shown in Figure 6.8. Users always have to start the transaction (i.e., insert their card and PIN number) and then 'operate on default' before proceeding to take their card, receipt and cash. The task is no more complicated on the occasions that people want the default amount shown, but the transitions through the task structure now require them to make a visual transition to the amount, whether they want the default or not. Once they have done this, they are more likely to 'notice' whether it is the amount that they want, and so will be less likely to slip into accepting it when it is not.

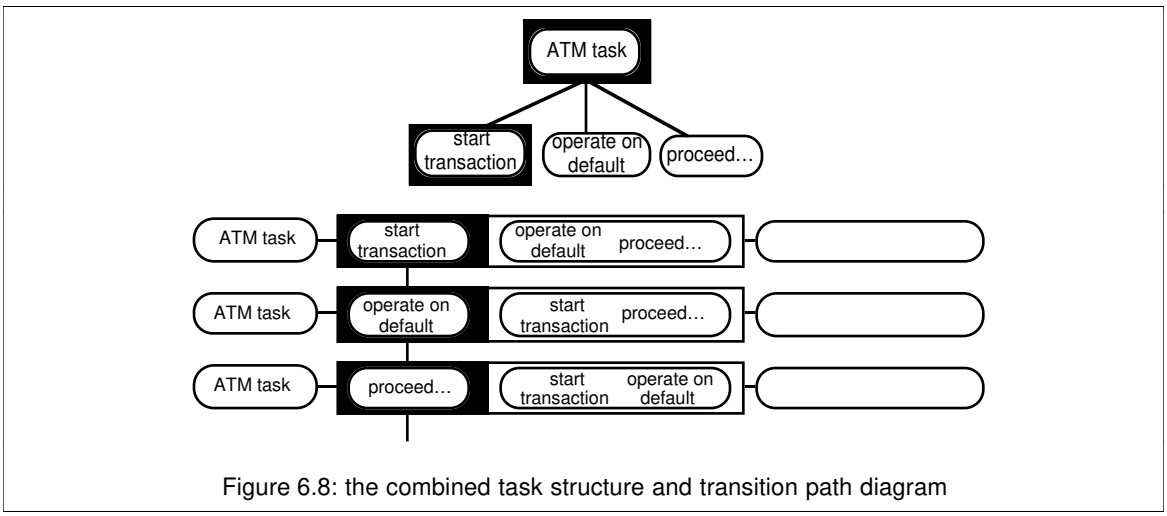

Figure 6.8: the combined task structure and transition path diagram

Complex multiwindow displays

The displays we have looked at in the examples so far have been comparatively simple, with only a few objects or groups of objects on the screen at a time. Many displays are, of course, more complex than this, particularly those with several different windows, which can be rearranged or resized by the user. Letting the user position groups of objects on the display clearly reduces the designer's ability to control the structure, but the techniques that we have presented can still provide some support. In particular, the contents of different windows can be compared with the tasks that the user will be carrying out, to ensure that they can easily move from window to window without having to laboriously search for the information that they want.

The interface shown in Figure 6.9 is a system that lets travellers make enquiries about internal flights in America. Travellers can use a mouse and keyboard to enter information, and can also input speech by holding down a button (The microphone icon in the 'Record' window). The translated speech input is shown in the 'Recognition' window. The multimedia aspects of this interface are not at issue here. We are going to consider the visual structure of the interface and think how this might affect the user's tasks.

Figure 6.10 shows the first level of a structural diagram for this screen. Because all of the windows can be moved around, resized, and repositioned by the user, we can't really tell anything about the groupings that they might form in practice (in Figure 6.9, for example, there is a cluster at the upper right, a cluster in the centre, and a group along the bottom of

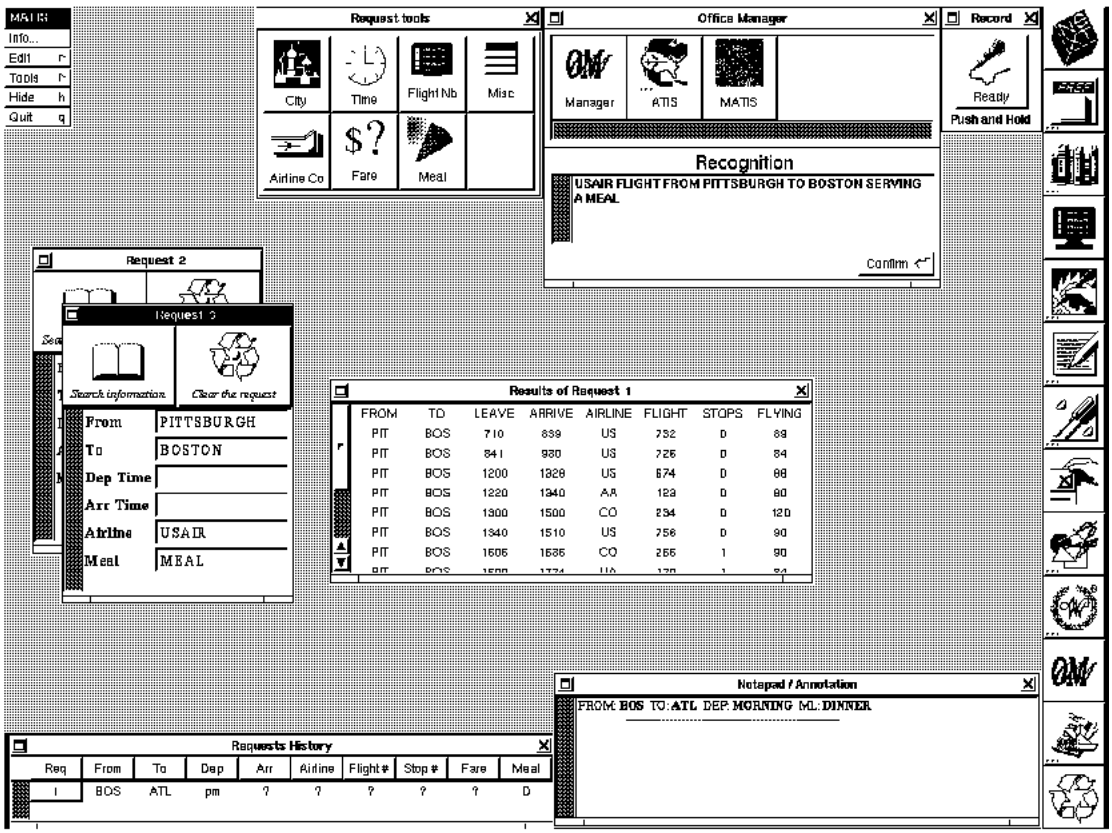

Figure 6.9: the MATIS screen display

the screen). The arrangement of the windows within the screen has been maintained in the structure diagram, to help you identify the screen objects that they refer to. This can be a useful technique for complicated displays, but it only really works for showing one level of a single object's structure at a time.

If we just look for now at the requests, which are on the left of the screen, we can produce a structure diagram of one of them (Figure 6.11). In this diagram we have included several levels of structure, and so have not been able to maintain the spatial organisation of the objects, as we could in Figure 6.10. We have been able to indicate which object within each level, if any, forms a pragmatic subject.

The window consists of the heading (which contains the name of the request and a close box), the icons (one to start a search, the other to clear the form), and the request form itself. This has a scrollbar and a list of search criteria, that each have a title and a slot, which will all be empty when a new request is created, but which will be filled in by the user. In this Figure we've just shown the structure of the 'From' and 'Arr Time' slots, but the others have similar structures.

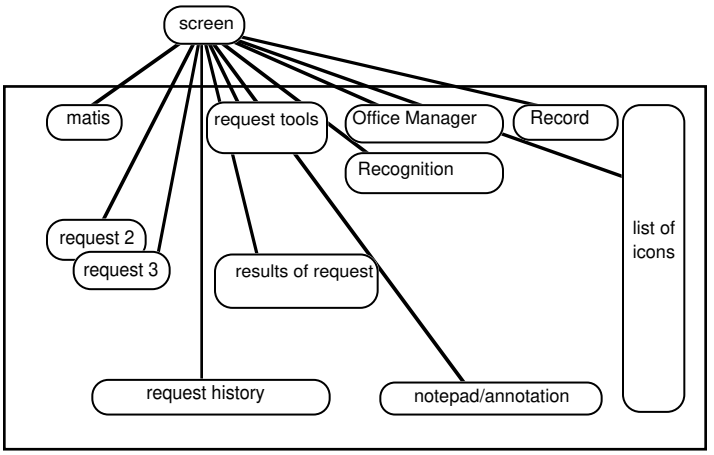

Figure 6.10: the top-level structure of MATIS, treating each window as a unit within the screen.

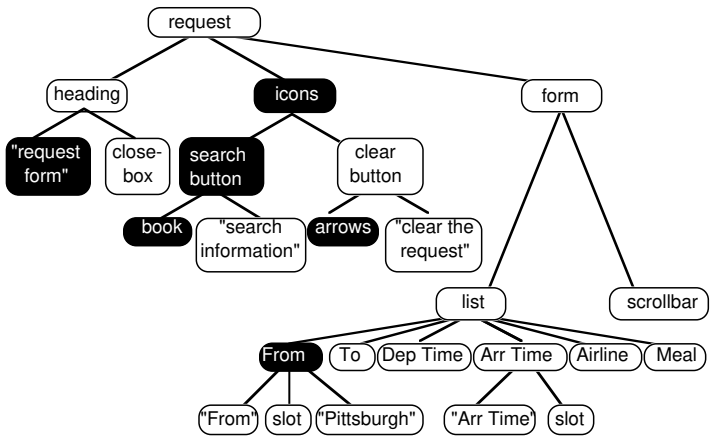

Figure 6.11: the structure of the MATIS request form.

What can you tell about the use of this structure? To begin with, the icons are likely to form the pragmatic subject of the form's structure, since as a group they pop out from the textual content of the rest of the objects. This is good, because in searching for this particular form, this group discriminates it from all other windows within the display – none of them contain this group or the two objects it is composed of. The user will be able to form the mental image of, say, 'a book', and will be able to reject any other windows as soon as they look at them.

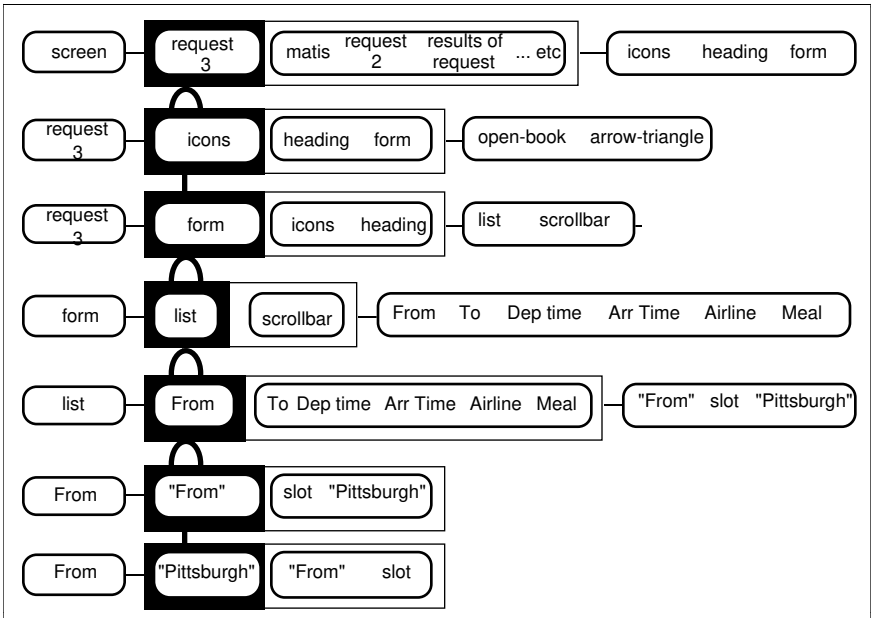

Figure 6.12: a transition path for the use of the request form

Once they have found this window, they will probably want to find the slot that they have to enter a specific piece of information in. Most requests will be to find a flight from one city to another. These two slots are the ones that are most likely to be filled first on a form. The names of these two slots appear at the top of the list on the left of the window, and since people usually read from left to right and from the top of a column down, we can suggest that the 'From' slot will be the pragmatic subject of this group, and so the easiest one to find – consistent with the users' most frequent task.

The 'granularity' that you need to use to describe a structure depends upon the way that people will have to use it. As we have seen with icons, the structure of objects might need to be considered if they have to be discriminated from one another, but it may be sufficient to describe, for example, three lines just as 'lines' without further decomposing them into 'line + line + line'. In Figure 6.11, you'll see that we haven't decomposed words into their constituent letters, nor the 'book' and 'arrows' icons into their structures, since in both cases the pragmatic subject of each of them would be sufficient to discriminate it from the other objects within its group. If you didn't know what these icons meant, though, and were searching for a textual label that meant 'start a search', you might have difficulty locating it, since it is part of the predicate of an icon, whose pragmatic subject (a book) is not usually associated with the task of searching.

Figure 6.12 shows a transition path that a user might make to fill in a request form (that is already open), to enter the information about the city they want to fly from. We can tell from the structural description that people need to read down the list of slot-names, and then make a transition to the slot. One thing that might make it easier for this last transition to be made is if the slot names were right justified, rather than left justified, but this in turn might break up the visual structure of the list, and make the name harder to find in the first place.

Looking at the items within the list, you can also see that in an empty form, they all have the same structure. As soon as they get filled, their structure becomes more complex - an additional part is added. As with the 'circles' and 'circles plus gaps', the filled slots stand out from the empty slots, and so are more noticeable.

Exercise Set 7

1 Draw structure diagrams for the 'Request Tools' and 'Results of request' windows shown in Figure 6.9 – but don't go into too much detail.

2. Complete the structure diagram of the Requests window by adding the contents of each filled and empty slot.

3 Suppose the user had just entered 'Pittsburgh' into the 'from' slot of the Request form, and 'Boston' into the 'to' slot, so that they are now looking at the word 'Boston' (and the rest of the slots are empty). Draw a transition path diagram to show how they would:
a) locate the search button to carry out the request
b) find a flight that departs before midday from the 'results of request' window (which will be the one shown in Figure 6.9).

4 Consider the task steps the user will be making at each moment in this transition path diagram (that is, before making each transition), and so what propositional representation will be guiding their search.
At one point the object representation provided by the display does not correspond to the propositional representation. Can you spot a simple change that you would recommend to improve this display?

5. Suppose all of the slots had been filled in – does this change the way the 'list' object is structured? Draw a new structure diagram for the 'list' object.