Chapter

11

Splitting Up Charts

The three graphical languages described in this book allow the decomposition of elements: each activity, state, or module is either basic or is described by a set of subelements. Other modeling notations and tools also allow multilevel descriptions, but many of them require that each level be described in a separate chart. Our languages allow multiple levels in the same chart but also allow the description to span several charts. In this chapter we discuss the possibility of presenting different levels of decomposition in separate charts. We deal mainly with linking the graphical information. The visibility of elements belonging to different charts is discussed in Chap. 13.

It is worth distinguishing separate charts depicting different levels of the decomposition from generic charts that are considered reusable components of a model. This chapter deals with the former; the latter are described in Chap. 14.

11.1 Separating a Chart into Multiple Pages

The charts drawn in earlier chapters contained a top-level box, representing the element being described. This box was then decomposed into lower-level boxes, with each level being drawn inside the higher one. See, for example, Figs. 2.3, 4.11, and 9.1. Often, however, it is convenient to break down the drawing into a number of charts, each containing one or more levels of decomposition. For example, instead of chart `A` of Fig. 11.1*a* we might want to draw the two separate charts of Fig. 11.1*b*. Although there are now two physically distinct charts, `A` and `A2`, logically there is just one, and the information in chart `A2` is treated as if it were drawn inside the box named `A2` in `A`. Thus there is a single *logical chart* that consists of two *physical charts,* which are also called *pages.*

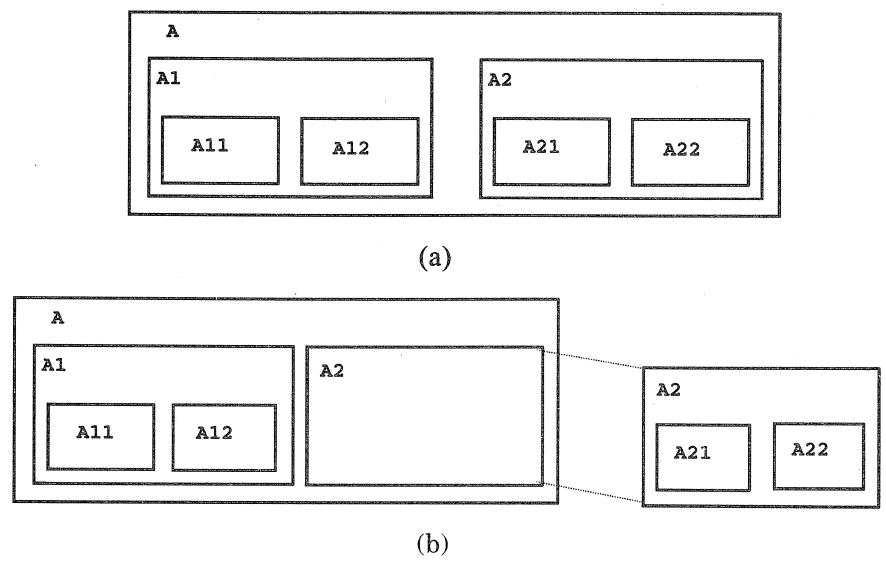

Figure 11.1 Splitting up a chart into pages.

Here are some of the reasons for dividing a chart into pages. They are similar to the reasons for breaking down a large piece of software into functions and subroutines.

- *Overly detailed charts.* A complex chart that contains too many details is difficult to read and comprehend. Breaking it down into several pages has a "decluttering" effect. Because this is the primary reason for dividing charts, we often term the separation of charts into pages as *decluttering*.
- *Information relevant to different people.* Often, different pieces of information in a chart are relevant to different observers. Here, the breakup is according to the responsibilities or interests of different people. We might call this *person-oriented information hiding*; that is, each person gets to see only the information relevant to the parts of the system he or she is working on. This is a widely acclaimed principle of system development, and decomposing charts into pages can help support it. Also, such a division can help overcome difficulties that arise when different people update parts of the same chart, or when one updates it while another analyzes it.
- *Information relevant to different levels.* Here, the idea is to support information hiding in the classical sense, that is, to make sure each level of the specification contains only the elements relevant to it.
- *Information from different configuration management units.* Here, the splitting is done according to different versions or releases (or

both) of the system under development, or according to different ownership and read/write/modify privileges.

- *Hybrid process of building the charts.* Some charts are built partly by a top-down process and partly by a bottom-up one. Breaking down charts can be used to draw the low-level components on separate pages and incorporate them as the internal descriptions of components in charts of higher levels. This introduces flexibility into the chart-building process.
- *Easing modification.* Splitting up the model into many charts can simplify the logistics of modification. Subcharts represented by separate pages can be replaced easily by others with the same interface. This makes it easy to present specification alternatives simply by changing the contents of black boxes.

Although chart decluttering can be beneficial, sometimes it is not recommended. We have in mind situations in which the system does not lend itself to neat structuring or where despite the availability of a good structuring there is a tight interrelationship between the low-level elements in different parts of the structure. In such cases, decluttered charts may be harder to comprehend. For example, it is sometimes easier to follow the behavioral aspects of a complex model when these elements are concentrated in a single statechart. The same goes for presenting and comprehending the flow of information in an activity-chart down to the basic low-level activity that actually produces and consumes the data elements.

11.2 Offpage Charts

We now discuss the mechanism used to split a chart into several pages. The contents of a box element (activity, state, or module) may be drawn in a separate chart. The box element is called an *instance box,* and the associated chart is called an *offpage chart* or a *definition chart.* The relationship between the two is sometimes termed the *box-is-chart* relation. The chart of the instance box is sometimes referred to as the *instance chart.*

To represent the relation between an instance box `A` and a definition chart `B`, we label the instance box by `A@B`, which means that this is box `A` but its internals are to be described in chart `B`. If we want to use the same name for the box and its definition chart, we may simply omit the first of the two names. Thus a box labeled `@A` means that the box and its definition chart are both named `A` (which is therefore like labeling it `A@A`).

In our EWS example, the functional decomposition of the `SET_UP` activity of Chap. 2 may be described in a separate chart. Figure 11.2*a* shows this activity named `@SET_UP`, meaning that its contents are

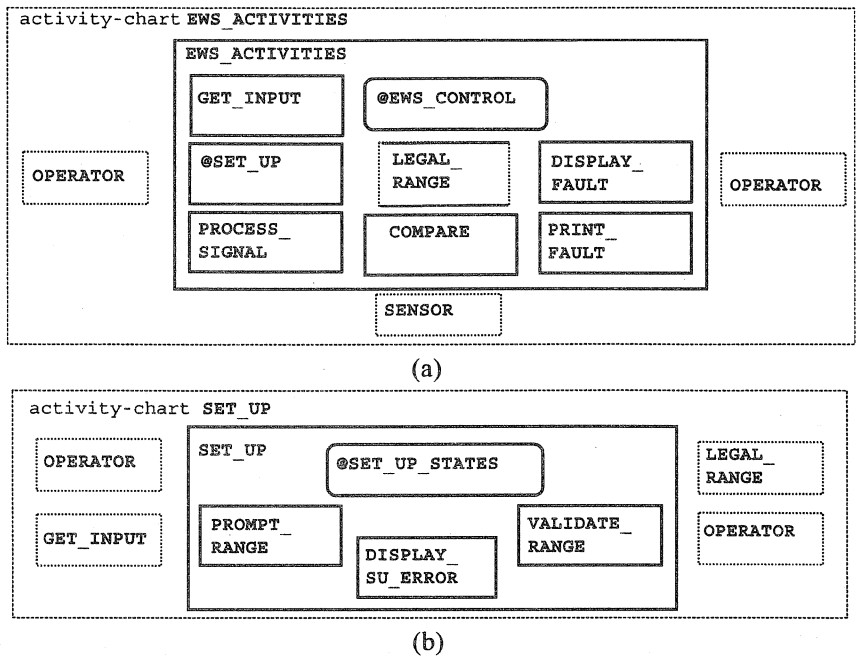

Figure 11.2 An instance activity and its definition (offpage) chart.

defined in a chart named `SET_UP`, and Fig. 11.2*b* shows the corresponding definition chart with its further decomposition. As explained earlier, no name precedes the `@` symbol, so the box name is the same as the definition chart name, and we may, for example, use the action `start(SET_UP)` in the controlling statechart `EWS_CONTROL`. Had we labeled the box `SU@SET_UP`, that action would have had to take the form `start(SU)`.

Note that the notation used to associate a box with its offpage chart is the same as that used to associate a control activity with its describing statechart. See Fig. 11.2*a*; the control activity labeled `@EWS_CONTROL` is described by a statechart named `EWS_CONTROL`.

When a box is described by an offpage chart, say, `A@B`, the definition chart `B` must have a unique top-level box, and the instance box `A` may have no subboxes. Of course, the subboxes appearing in the top-level box in `B` are considered *logical subboxes* of `A`, but `A` has no *physical subboxes*. This terminology is used for parents, too. Boxes may thus have *logical* and *physical parents*.

Referring again to Fig. 11.2, the `PROMPT_RANGE` activity is considered a subactivity of the instance activity `SET_UP`, and therefore it is also a logical descendant of `EWS_ACTIVITIES`. The physical parent of `PROMPT_RANGE` is the top-level activity `SET_UP` in the activity-chart with the same name. Because the top-level box is considered an "image" of the instance box, we have named the two identically in our example. However, it is possible—although not recommended—to

have three different names, one each for the instance box, the definition chart, and the top-level box.

The external activities that are presented in the definition chart are the boxes that surround the SET_UP activity in the instance chart (EWS_ACTIVITIES) with which SET_UP communicates. We shall return to this issue in the following section and in Chap. 12, where the entire model is discussed.

Both the instance box and the top-level box of the definition chart have associated entries in the Data Dictionary, and the information appearing therein must be consistent. More specifically, the following fields, if not empty, must contain the same information: Termination Type and Implemented by Module in an activity entry and Described by Activity-Chart in a module entry. For all other fields, such as Static Reactions and Active Activities in a state entry and Attributes for all elements, the information in the entries for the instance box and the top-level box of the definition chart is accumulated and viewed as applying to the common entity.

We do not allow multiple instances of a common definition chart. In other words, two instance boxes cannot be described by the same definition chart. When the need arises for multiple instances of the same chart, the generic chart mechanism of Chap. 14 should be used.

11.3 Connecting Offpage Charts by Matching Flows

One advantage of having multiple levels in the same chart is the ease of viewing arrows (flow-lines in activity-charts and module-charts, and state transitions in statecharts), in that sources and targets are seen together. When charts are decluttered into separate pages, this will necessarily be less convenient. In any case, we need to have reasonable mechanisms for combining arrows over pages. We supply two. The first, discussed in this present section, concerns matching flows and can be used only in activity-charts and module-charts. The second concerns diagram connectors and is described in Sec. 11.4. Although diagram connectors can be used in all three types of charts, we describe their use for statecharts only, because the first method is preferred for the two other types of charts.

Here is how to link flow-lines between pages in activity-charts and module-charts. The arrows leading to and from the borderline of the instance box are matched with the arrows exiting or entering external boxes in the definition chart. The actual matching is carried out by identifying common information elements included in the labels.

Let us examine an example. In Fig. 11.3 part *a* is the original chart and *b* describes its partition into two charts, by extracting the contents of A1 and relegating them to a new activity-chart. The flow-lines in activity-chart A that depict the interface of activity A1 are all connected to the borderline of the instance box, including those that are related to

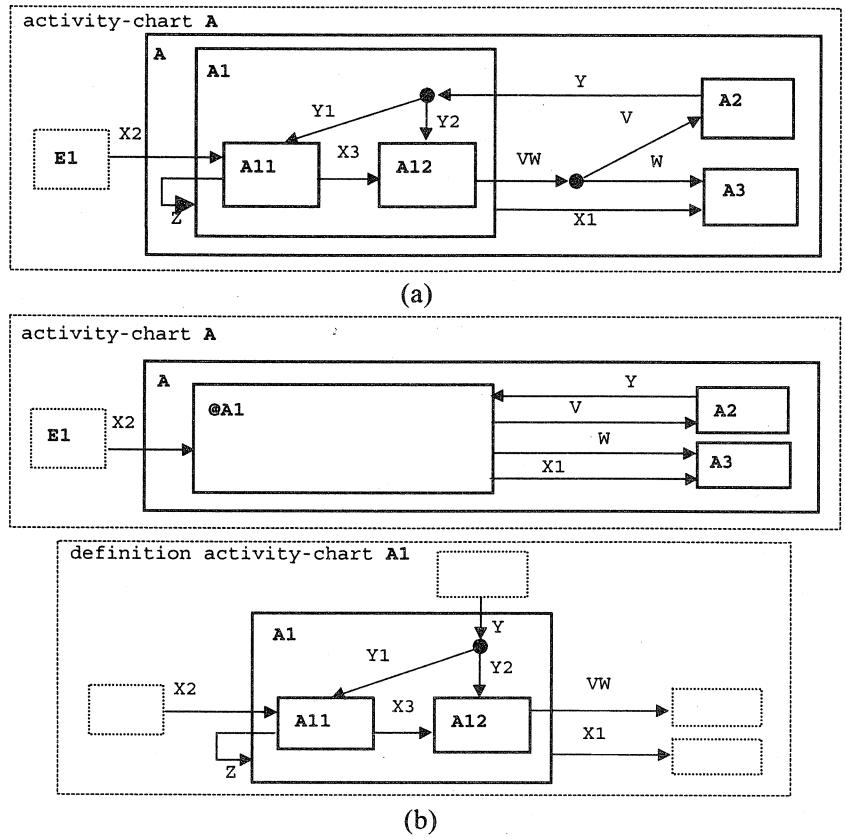

Figure 11.3 Connecting pages by matching flows.

the internal activities of `A1`. In the definition chart of `A1`, all flow-lines are labeled with the flowing elements and are connected to their actual sources and targets inside `A1`.

Note that the matching is carried out according to the flowing elements and not the written labels. For example, in the definition chart `A1`, the flow-line emanating from `A12` is labeled `VW`, an information flow consisting of `V` and `W`. This line is matched with the two separate flows labeled `V` and `W` in the instance chart.

Note also that the external boxes in the definition chart of Fig. 11.3*b* are unnamed. This is done mainly to emphasize the fact that arrows are linked by matching the flowing elements and not by the sources and targets. However, the names may be added if it is important to represent these sources and targets explicitly. This indeed might be the case in a top-down development effort because the sources and targets are already determined in the instance chart.

Another point worth making is that there is no correspondence between sources and targets of the flows in the definition and instance charts. For example, `V` and `W` of Fig. 11.3, when considered as

the compound information item VW, have a single external target in the definition chart, whereas in the instance chart they lead to two separate boxes. This illustrates the fact that unnamed external boxes are really just place holders, or external agents that are connected to arrows that lead to or from the outside. (In a bottom-up development effort, this is particularly helpful; we might not want to specify the actual external elements when developing the definition chart because we might not yet know about them.) If the external boxes in the definition chart *are* named, the names must be consistent with those of the corresponding sources and targets in the instance chart.

For example, Fig. 11.4 shows the SET_UP definition chart with its external interface. Comparing it with Fig. 10.5, we see that the boxes in this external interface correspond to the various boxes with which the SET_UP activity communicates. In the case of decluttering an activity-chart, the external activities in the definition chart may correspond to the following kinds of elements in the instance chart: regular internal activities, control activities, external activities, and data-stores. In particular, the data-store LEGAL_RANGE is also depicted as an external activity in the definition chart. In a similar way, when decluttering a module-chart, external modules in the defining chart may correspond to execution modules, storage modules, or external modules in the instance chart.

Clearly, each input or output of the top-level box in the definition chart must also appear in the instance chart, either as a direct flow to the instance box or as a flow-line connected to one of its ancestors. We also expect each flow-line connected to the instance box to appear in the definition chart that contains the particular source or target, even when it is specified as being consumed or produced by all subelements of the instance. For example, comparing Fig. 11.3*b* with Fig. 11.3*a*, we see that although X1 is an output of A1, it also appears in the definition chart. The reason for this is that when drawing the interface of the instance, it is considered as the interface to a "black box." That is, drawing an input line means "one or more of the components consume this input, and the actual consumer(s) will be specified in the definition chart." The reasoning is similar for outputs.

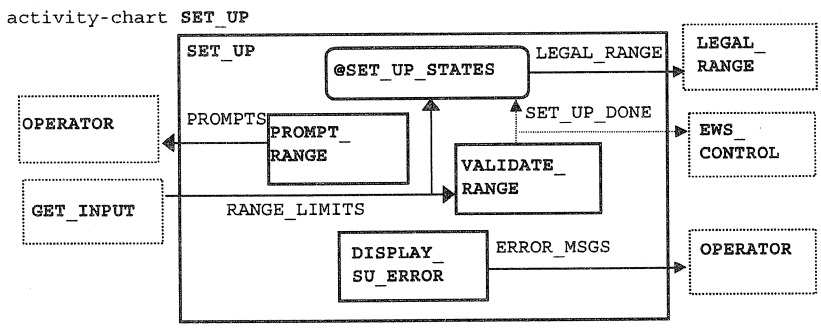

Figure 11.4 SET_UP definition chart.

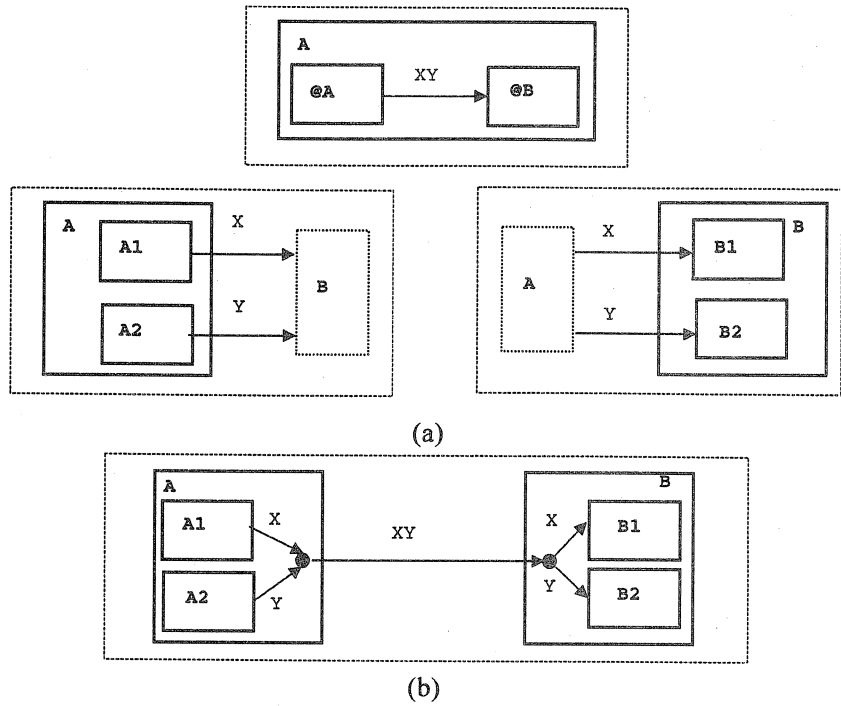

Figure 11.5 Compound flow-lines distributed over several pages.

In Sec. 2.5.4 we introduced the notion of a compound flow-line; we talked about the logical flows between activities (or modules) that consist of several flow-line segments linked with connectors. Now here, although using a different construction method, we have compound lines that are distributed over several pages. Figure 11.5*a* shows an example that contains two compound flow-lines: `X`, flowing from `A1` to `B1`, and `Y`, flowing from `A2` to `B2`. An equivalent construct is shown in Fig. 11.5*b*.

11.4 Connecting Offpage Statecharts Using Connectors

The method presented above for connecting offpage charts to the description in the instance chart cannot be applied in the case of statecharts because they are not connected via flows. For them we use an alternative mechanism that is based on diagram connectors. In earlier chapters we already used diagram connectors to combine several arrow segments into a single logical compound arrow. See, for example, Fig. 4.21, in which three compound transitions between states were constructed from two segments each, using diagram connectors. Because these connectors appear in the same chart, or page,

we refer to them as *inpage diagram connectors.* When they are used to connect arrows on separate pages, as is the case here, we call them *offpage diagram connectors.* In the instance chart (i.e., the chart that contains the instance box) the connectors are drawn inside the instance box, and in the definition chart they are drawn outside the top-level box.

Like inpage connectors, offpage diagram connectors may be labeled either with numbers or with an alphanumeric string that starts with a letter and might contain underscores. A useful convention is to label the connector with the name of the source or the target of the arrow in the instance chart. Another possibility is to use the name of the trigger of the transition.

Each connector in the instance chart must have a matching connector in the definition chart, with consistent directionality of the arrow. See Fig. 11.6*b,* in which one arrow enters the GO connector in the instance chart and one exits the GO connector in the definition chart. A connector is not allowed to have both entering and exiting arrows. We allow several offpage connectors in an instance box, all with the same label, and follow a similar convention for connectors in the definition chart. Such multiple occurrences in one chart must all have the same arrow directionality. The same label can also be used for offpage connectors in separate instances. However, we do not allow an offpage connector in an instance box to have the same label as an inpage connector on the same page, because that might be confusing.

When imagining the compound arrows constructed from arrow segments leading to and from connectors, the offpage connectors are treated like junction connectors (as in the inpage case; see Chap. 4). Consequently, the triggers on these segments are combined by *and,* and all the actions on them are performed.

When connecting the statechart pages logically, the only transitions that have to be connected are those that cross the boundary of instance states. Transitions that enter or exit an instance state without crossing its borderline will typically not appear in the definition chart at all. The reason is that such entering transitions will enter substates in the definition chart via the default connectors, and the exiting transitions will exit the state regardless of the internal configuration. This rule is consistent with the idea of a structured specification, in that the reasons for entering and exiting the state are not to be known inside the state. Exceptions to this rule include exits that do not necessarily apply on the top level, that is, to all internal states, but only to some of them. In such cases, it is appropriate to describe the outgoing transitions in the definition chart as well as in the instance chart.

Figure 11.6 contains an example: part *a* shows the chart before decluttering, and part *b* after it. Notice that in Fig. 11.6*b,* transitions that cross the borderline of state ON are connected by connectors, while those

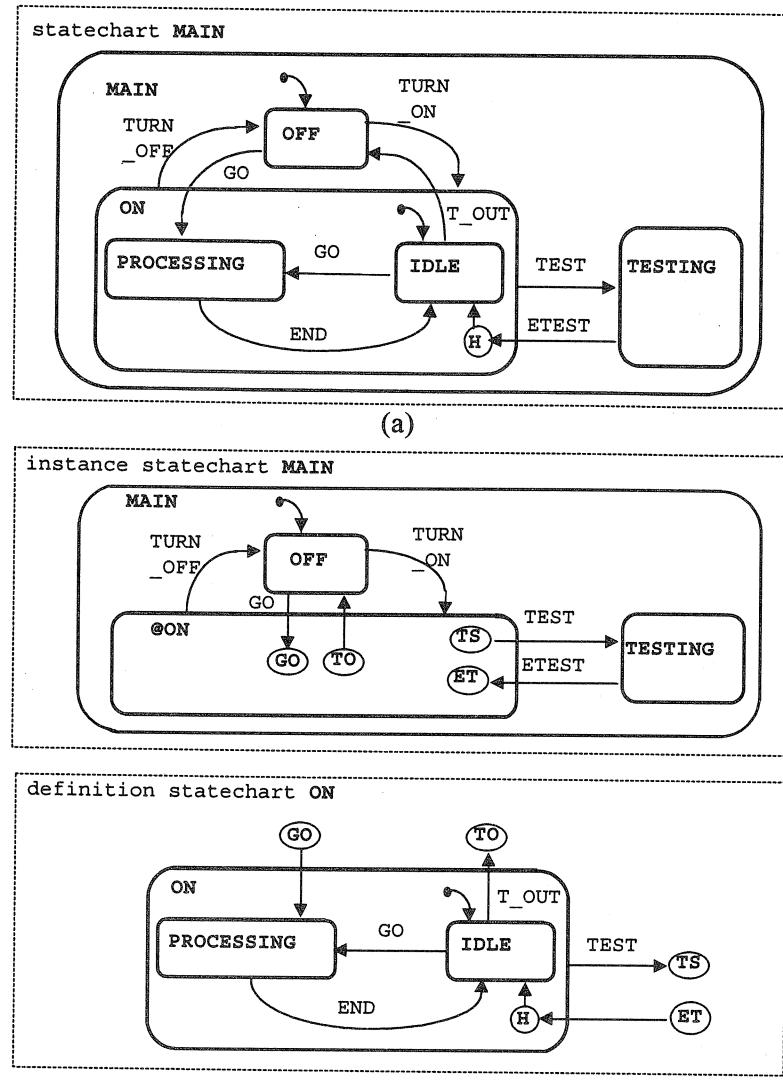

Figure 11.6 Transitions between pages of a statechart.

that emanate from that borderline (i.e., TEST and TURN_OFF) are drawn with or without the connector, depending on the particular case. The fact that TURN_OFF is an event that triggers an exit from every state is important information on the upper level. On the other hand, the decision about which states the event TEST acts on was made on the lower level. In Fig. 11.6*b*, the trigger labels appear in at least one page, depending on the specifier's preference, but not necessarily in both.