# Statemate Course

Statemate/SDL — Teleteaching Vorlesung

W.-P. de Roever
K. Baukus
CAU Kiel

D. Hogrefe
H. Neukirchen
MU Lübeck

# Session 3

# Activity-Charts

**Literature:** Chapters 2 and 3 of "Modeling Reactive Systems with Statecharts", by David Harel and Michal Politi. McGraw-Hill, 1998.

# 3 Describing the functional view of a system

- Activity-charts are used to depict the functional view of a system under development (SUD), "what the SUD does".
- This view of a system is specified by
  - a hierarchy of functional components, called **activities**,
  - what kind of information is exchanged between these activities and is manipulated by them,
  - how this information flows,
  - how information is stored, and
  - how activities are **started and terminated**, i.e., **controlled**, if necessary, and whether activities are **continuous**, or whether they **stop by themselves**.
- Activity-charts are kind of **hierarchical data flow diagrams**:

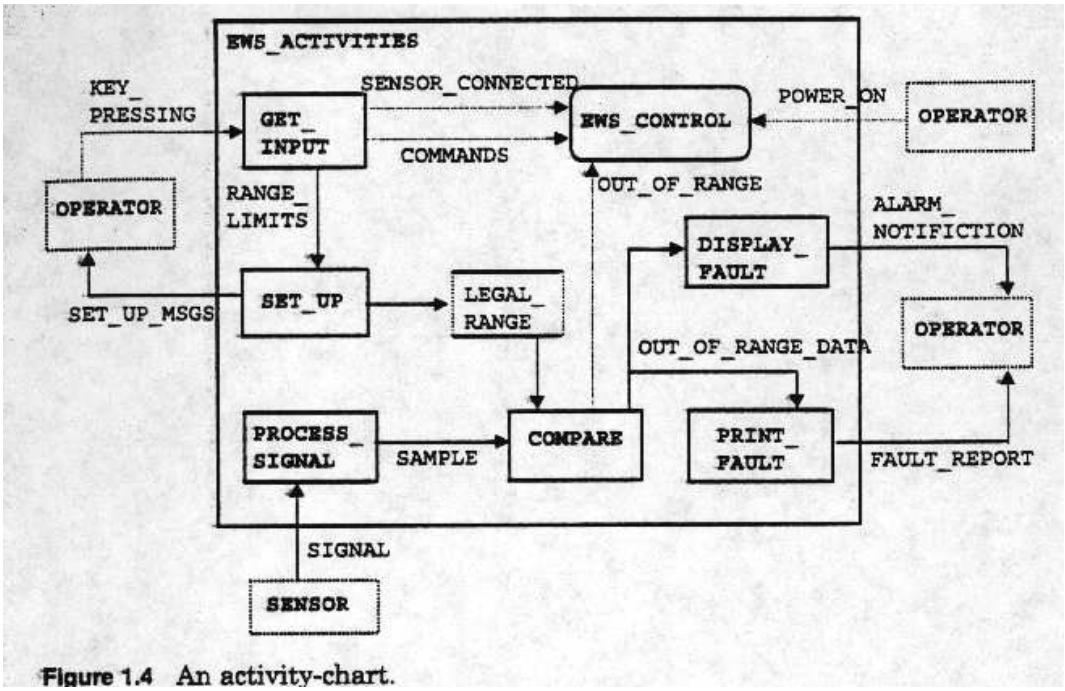



Figure 1.4 An activity-chart.

# 3.1 Functional decomposition of a System

The functional view of a system specifies the system's **capabilities**.

- It does so in the context of the system's environment, that is, it defines the environment with which the system interacts and the interface between the two:

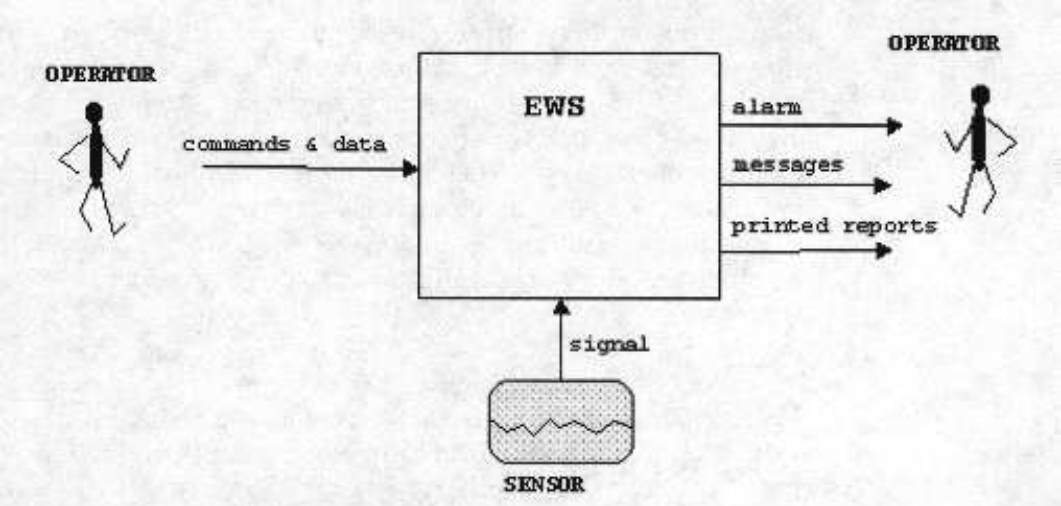



*Figure 2.1.* The context of the EWS

- This functional view does not address the physical and implementation aspects of the system; the latter is done in its **structural** view, i.e., its **module-chart**:

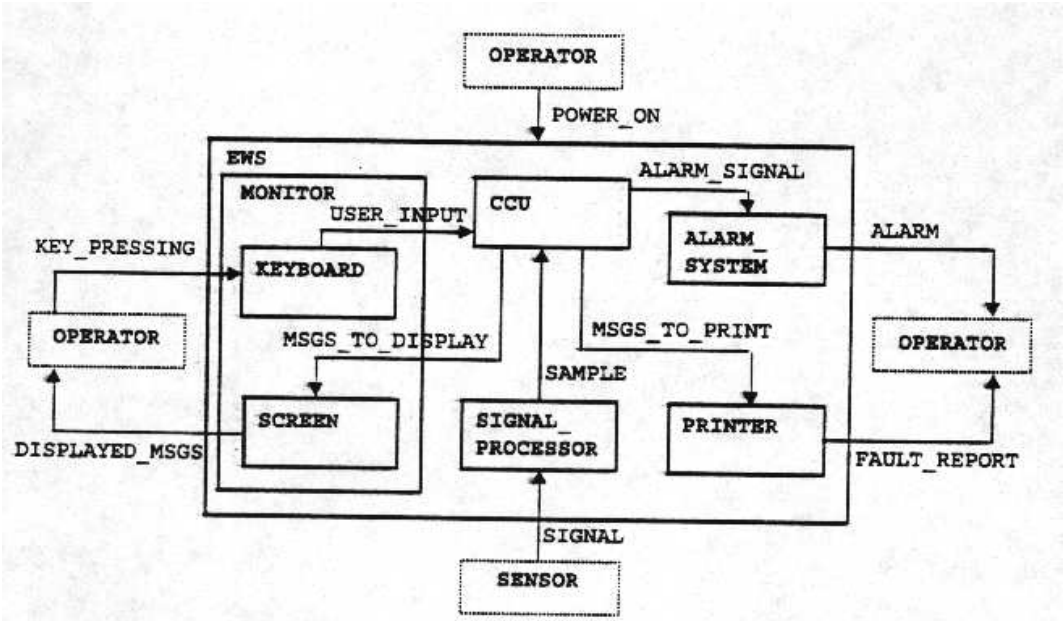



Figure 1.7 A module-chart.

- Moreover it separates the dynamics and behavioral aspects of the SUD from its functional description. The former is done by its **behavioral** view, in its controlling Statecharts:

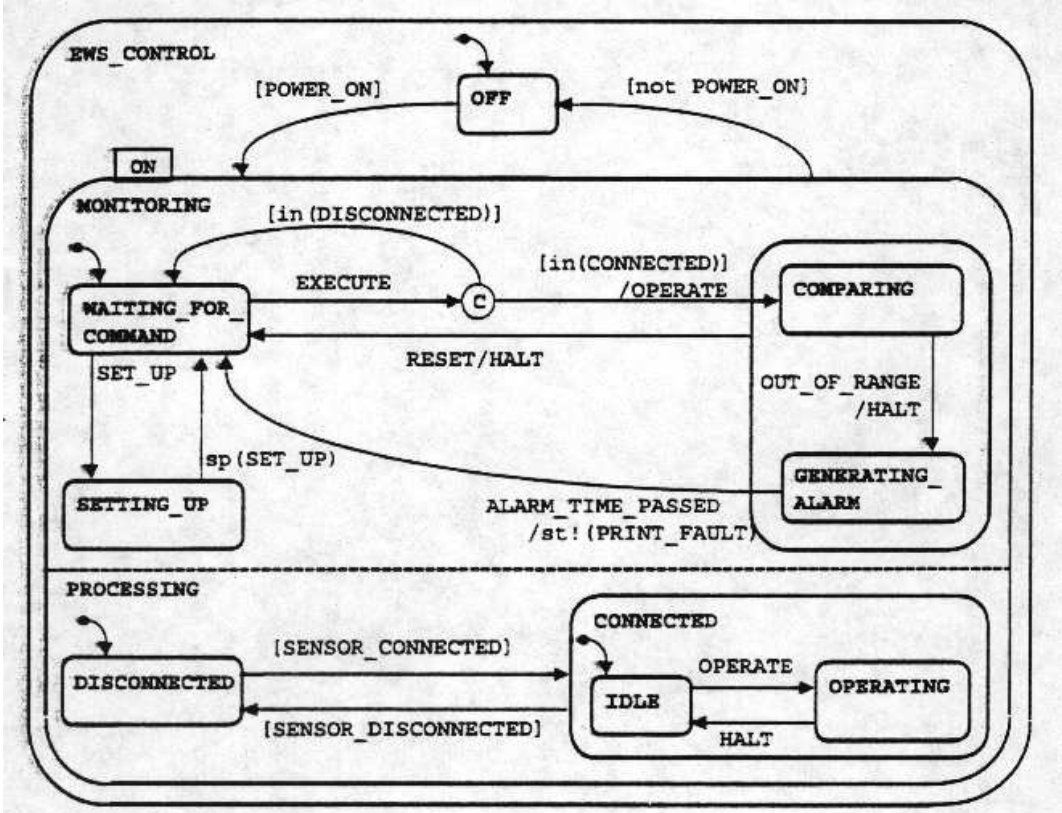



Figure 1.6 A statechart.

# Example

The **functional view** tells whether a medical diagnosis system can monitor a patient's functions, and, if so, where it gets its input data and which functions have access to the output data.

The **behavioral view** tells under which conditions monitoring is started, whether it can be carried out parallel to temperature monitoring, and how the flow of control of the process of monitoring develops.

The **structural view** deals with the sensors, processors, monitors, software modules and hardware necessary to implement the monitoring system

# The three views

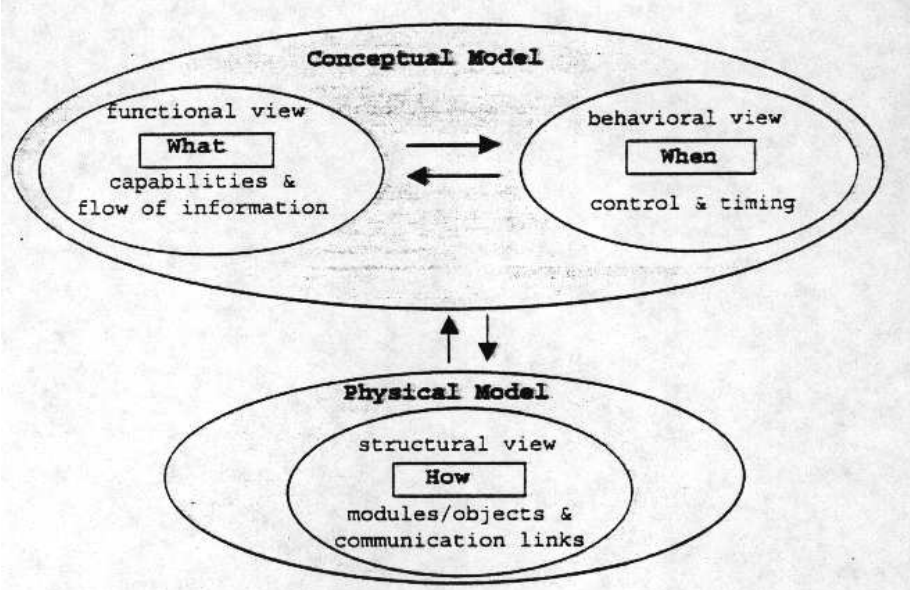



**Figure 1.2** The three specification views.

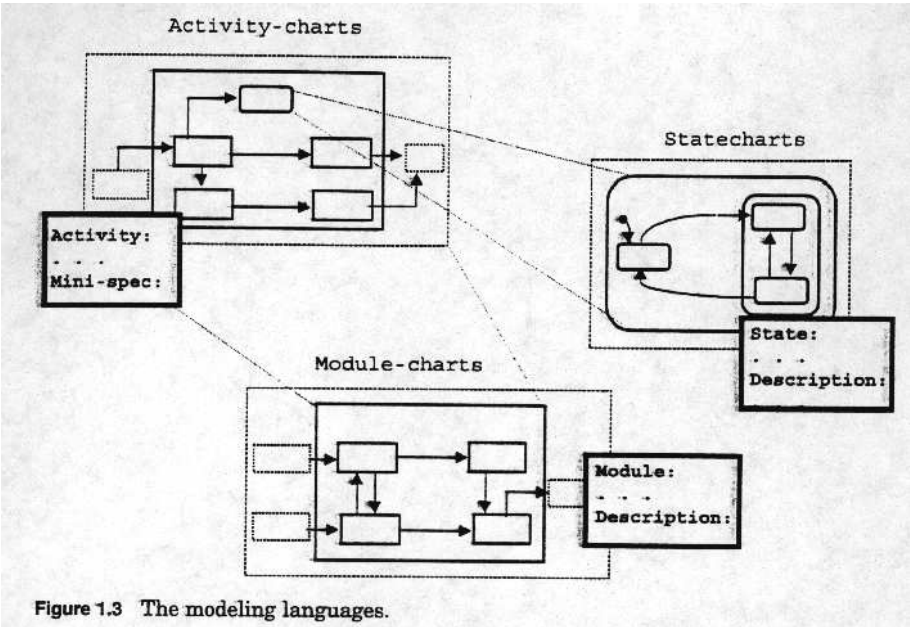



**Figure 1.3** The modeling languages.

# 3.1.1 Functional Decomposition

- In the Statechart approach, the functionality of a system is described by **functional decomposition**, by which a system is viewed as a collection of interconnected functional components, called **activities**, organized into a hierarchy.

- E.g., in the activity-chart EWS_ACTIVITIES, the SET_UP components can be decomposed leading to a multi-level decomposition of EWS_ACTIVITIES:

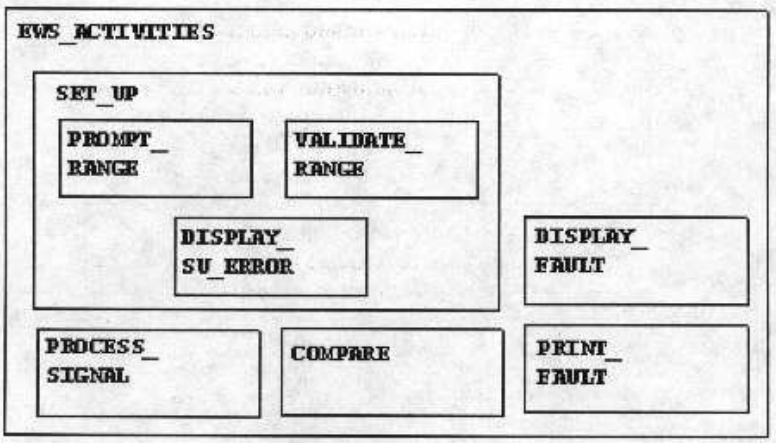



*Figure 2.3.* Multi-level decomposition of an activity

- Each of the activities may be decomposed into subactivities repeatedly until the system is specified in terms of **basic activities**.

  There are specified using textual description (formal or

informal), or code in a programming language, inside the Data Dictionary.

- The intended meaning of the functional decomposition is that **the capabilities of the parent activity are distributed between its subactivities**.

- The **order** in which these subactivities are performed, and the **conditions** that cause their **activation** or **deactivation** are not represented in the functional view and are specified in the **behavioral view**, i.e., in the (one) statechart associated with the parent activity-chart.

- Please observe that a functional component may very well be **reactive** in nature (cfr. the first session lecture).

- Activities can represent **objects, processes, functions, logical machines**, or any other kind of **functionally distinct entity**.

- In the following sections we'll confine ourselves to **function-based decomposition** of an activity-chart. We shall not discuss **object-based decomposition** (see Section 2.1.3 of Harel & Politi)

# 3.1.2 Function-based decomposition of activity-charts

- In function-based decomposition, the activities are (possibly reactive) functions.

- As an illustration consider the EWS example.

- Its first description is in natural language:

The EWS receives a signal from an external sensor. When the sensor is connected, the EWS processes the signal and checks if the resulting value is within a specified range. If the value of the processed signal is out of range, the system issues a warning message on the operator display and posts an alarm. If the operator does not respond to this warning within a given time interval, the system prints a fault message on a printing facility and stops monitoring the signal. The range limits are set by the operator. The system becomes ready to start monitoring the signal only after the range limits are set. The limits can be re-defined after an out-of-range situation has been detected, or after the operator has deliberately stopped the monitoring.

- Next we decompose this narrative to describe its functionality:

- The EWS receives a signal from an external sensor.
- It samples and processes the signal continuously, producing some result.
- It checks whether the value of the result is within a specified range that is set by the operator.
- If the value is out of range, the system issues a warning message on the operator display and posts an alarm.
- If the operator does not respond within a given time interval, the system prints a fault message on a printing facility and stops monitoring the signal.

- Thirdly, we identify the various functions that are described by there requirements:

**SET_UP**: receives the range limits from the operator.

**PROCESS_SIGNAL**: reads the "raw" signal from the sensor and performs some processing to yield a value that is to be compared to the range limits.

**COMPARE**: compares the value of the processed signal with the range limits.

**DISPLAY_FAULT**: issues a warning message on the operator display and posts an alarm.

**PRINT_FAULT**: prints a fault message on the printing facility.

- Notice that this description also contains info about handled data. An activity may **transform** its input into output to be consumed by other functions, which are **internal** or **external** to the system:

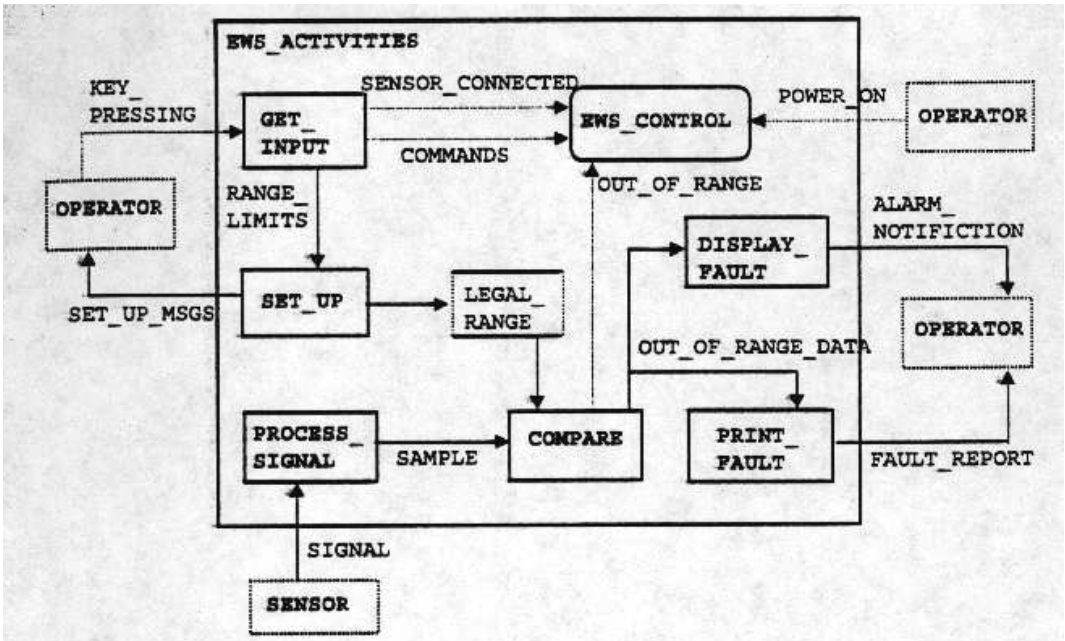



Figure 1.4 An activity-chart.

- The interface of an activity is described in terms of input and output signals, both data and control, see last figure.

# 3.1.3 System context

One of the first decisions to be made when developing a system involves its **boundaries**, or, **context**. I.e., one must determine which entities are part of the environment of a system, and how they communicate with the system. The latter are called **external activities** of the system.

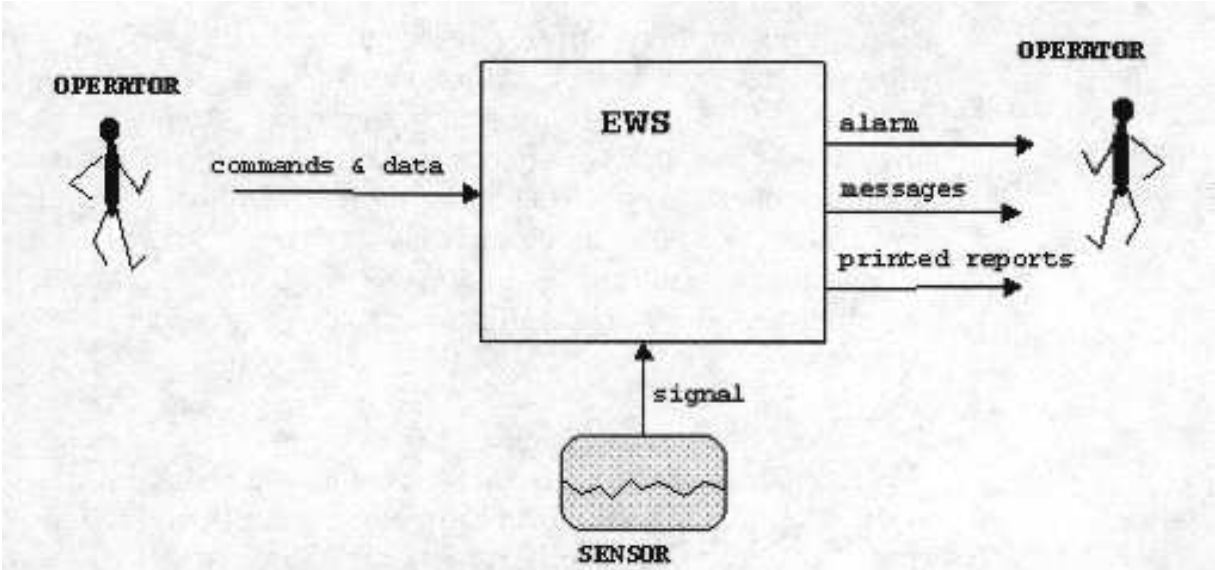



*Figure 2.1.* The context of the EWS

Notice that for the EWS one might have chosen for the printer to be external, leading to printer as external activity.

Different occurrences of the same entity (here: operator) denote the same entity; these are multiplicated of ease of drawing.

# 3.1.4 Decomposition process

The functional view is specified by **Activity-charts**, together with a **Data Dictionary** that contains additional information about the elements appearing in the charts, e.g., about their basic activities.

# 3.2 Activities and their representation

We continue the functional decomposition of the EWS, started with:

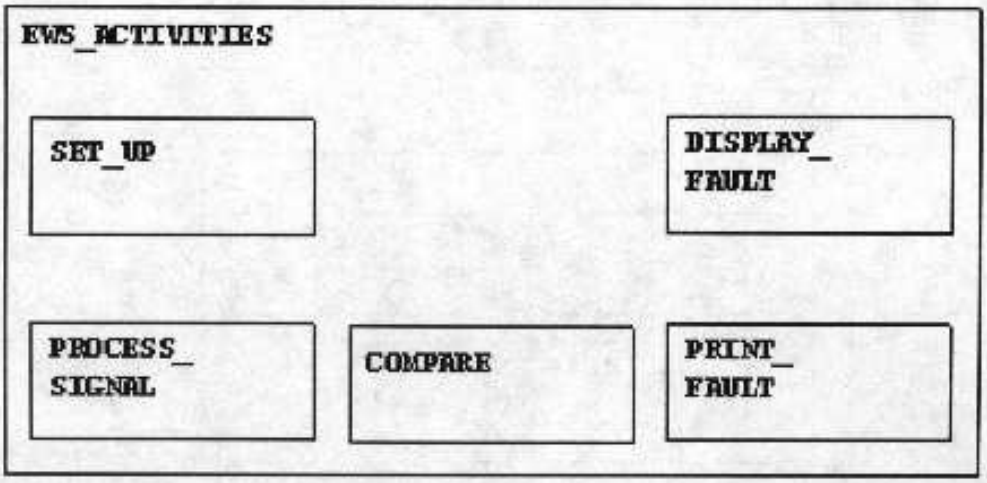



*Figure 2.2.* First level decomposition of an activity

This activity chart contains one top-level box, representing the **top-level activity** of the chart.

On their turn, the activities appearing above can be decomposed themselves, as SET_UP:

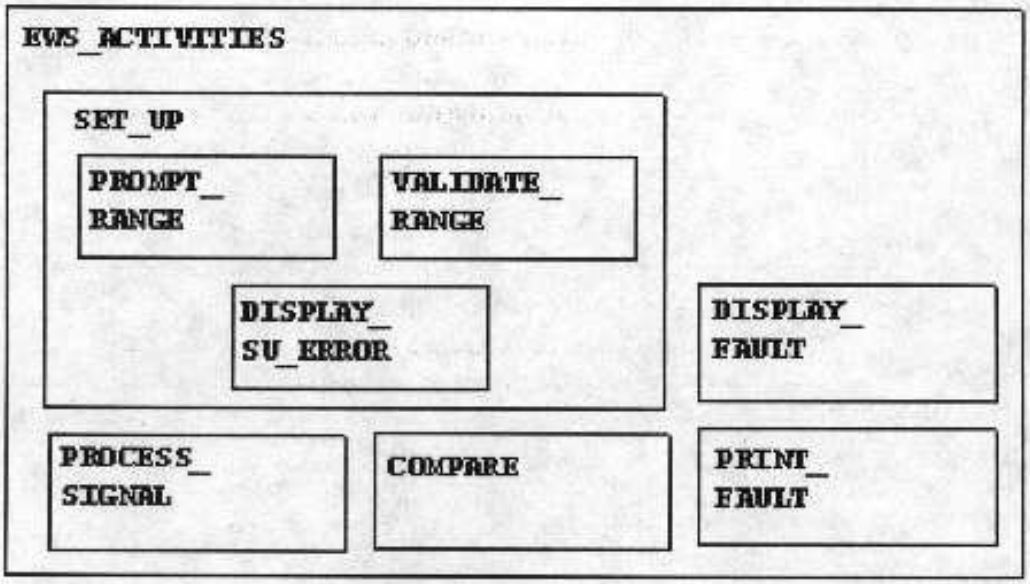



*Figure 2.3.* Multi-level decomposition of an activity

# Some terminology

- EWS_ACTIVITIES is called **top-level activity**
- EWS_ACTIVITIES is also called **parent activity** of SET_UP, COMPARE, etc., which are called **descendants** of EWS_ACTIVITIES, as are the subactivities PROMPT_RANGE etc. of SET_UP, who have SET_UP and EWS_ACTIVITIES as **ancestor**.

Each activity has a corresponding item in the Data Dictionary, which may contain additional information.

# 3.3 Flow of Information between Activities

- Consider the following chart:

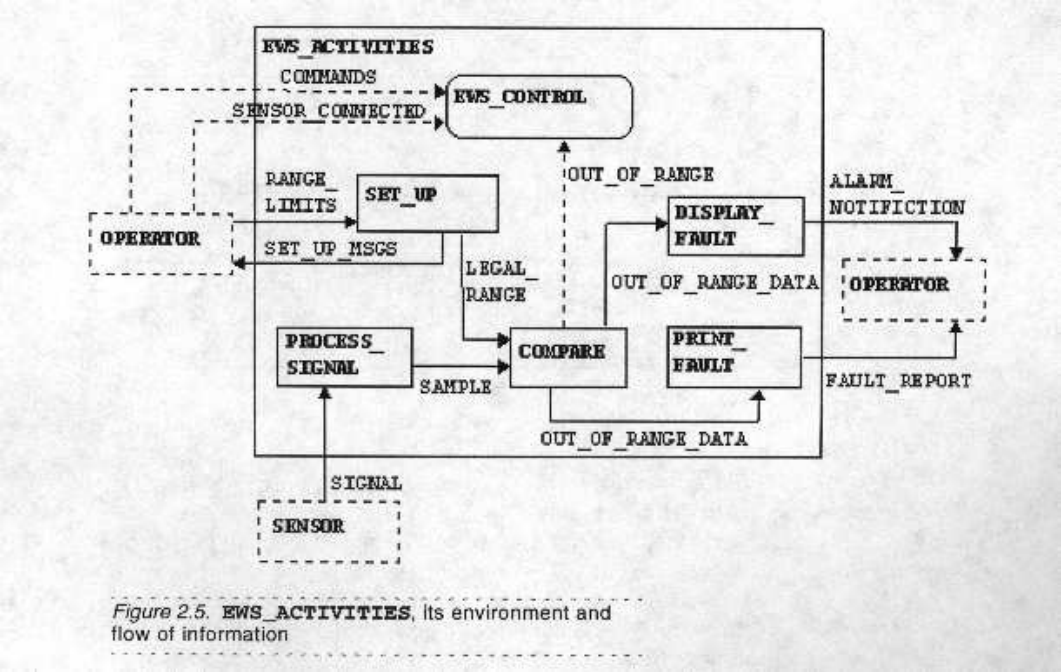



*Figure 2.5.* **EWS_ACTIVITIES**, its environment and flow of information

- OPERATOR and SENSOR are external activities, drawn using dotted lines.
- Different occurrences of OPERATOR refer to the same entity.
- Solid arrows denote **data-flow-lines** between activities.
- Control of EWS_ACTIVITIES is handled in its **control activity chart** EWS_CONTROL, a statechart (drawn using rounded corners).
- Dotted arrows denote **control-flow-lines**, carrying info or signals used in making **control decisions**.

# 3.3.1 Flow lines

A label on a flow line denotes:

- Either a single information element that flows along the line, i.e., a **data-item**, **condition**, or **event**.
- Or a group of such elements, as in, e.g.:

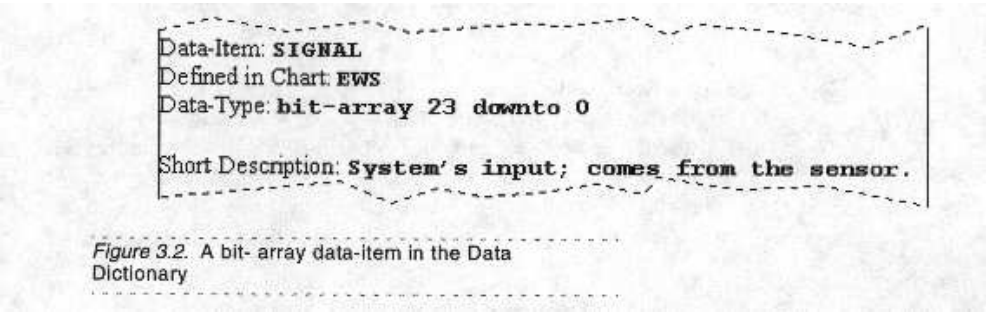



*Figure 3.2.* A bit- array data-item in the Data Dictionary

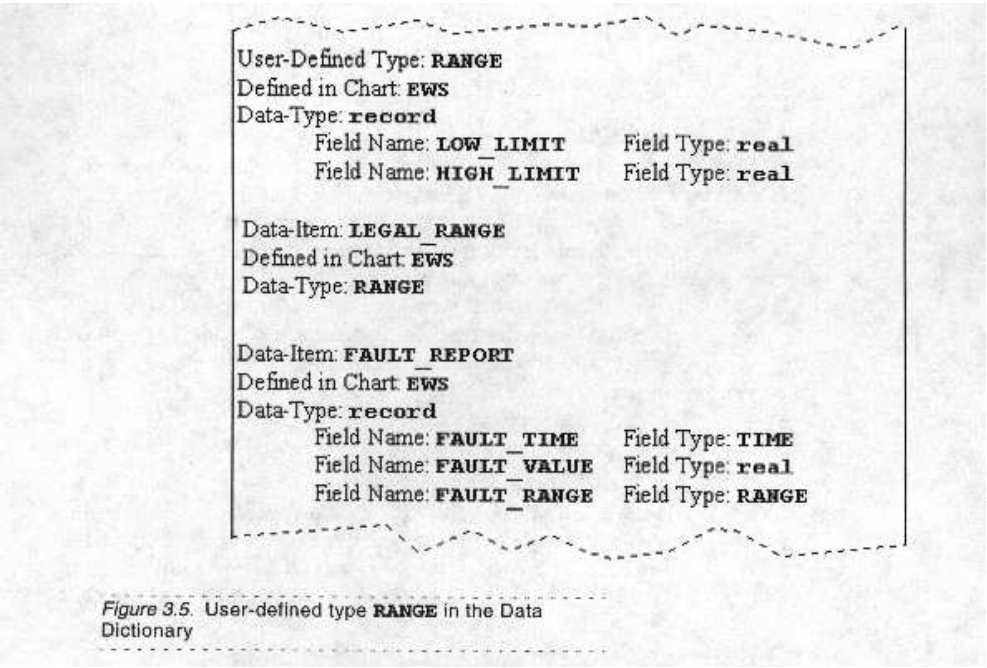



*Figure 3.5.* User-defined type **RANGE** in the Data Dictionary

Such a group is called **information-flow**.

- A flow-line originates from its **source activity**, and leads to its **target activity**:

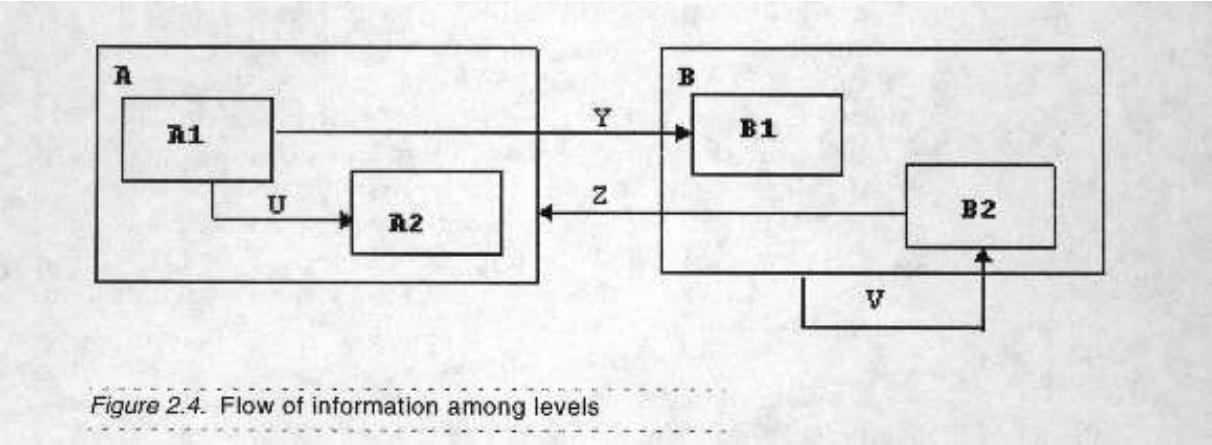



*Figure 2.4.* Flow of information among levels

- An arrow can be connected to a non-basic box, meaning it relates to **all the subboxes** within the box in question, see above the data flow lines labeled V and Z.

- Information flow SIGNAL in Figure 2.5 is declared in the Data Dictionary as in Figure 3.2 and is used in data processing.

- Information flow COMMANDS in the Data Dictionary declared as below, is used to denote **control issues**.

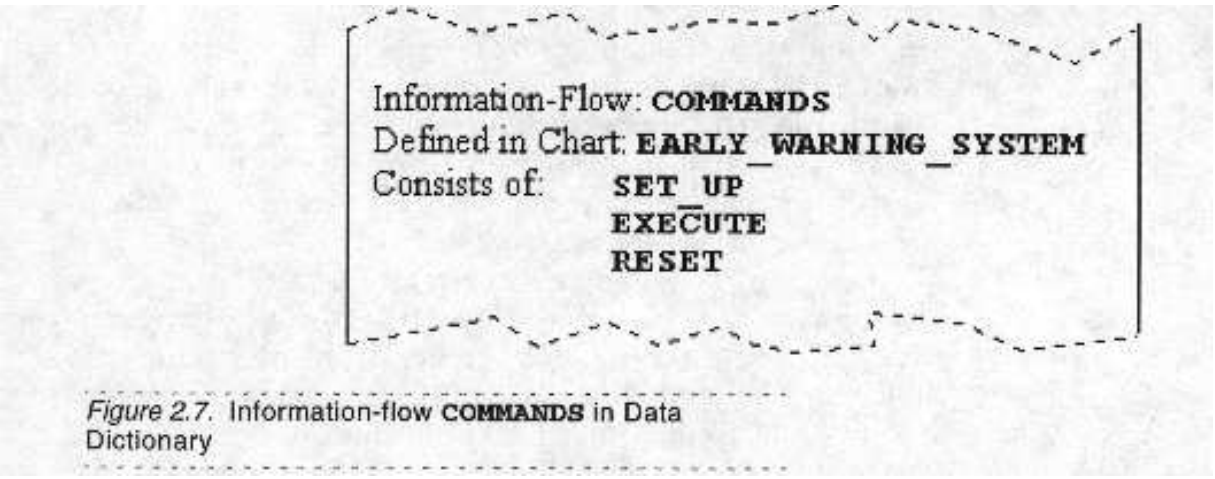



*Figure 2.7.* Information-flow **COMMANDS** in Data Dictionary

- Flow lines may represent, e.g.,
  - parameter passing to procedures
  - passing of values of global variables
  - messages transferred in distributed systems
  - queues between tasks in real-time applications
  - signals flowing along physical links in hardware systems

- Flows can be continuous or discrete in time.

# 3.3.2 Flowing elements

- Three types of information elements flow between activities: **events, conditions, data-items**.
- Their differences are in their **domain of values** and **timing characteristics**:

  **Events** are instantaneous signals used for synchronization purposes, e.g., OUT_OF_RANGE in Figure 2.5.

  **Conditions** are **persistent** signals that are either true or false, e.g., SENSOR_CONNECTED in Figure 2.5.

  **Data-item** are **persistent** and may hold values of various types and structures, e.g., SIGNAL, a **bit-array**, or LEGAL_RANGE, a**record** with two fields of type **real**, HIGH_LIMIT and LOW_LIMIT.
- All three types of information elements can be arranged in array and record structures:

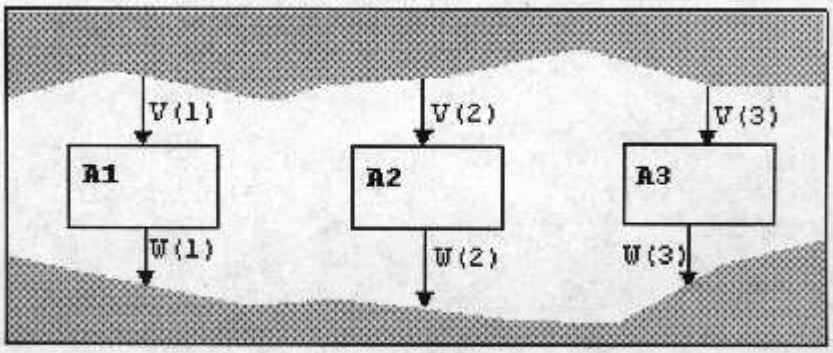



*Figure 2.6.* Array components labelling flow-lines

# 3.3.3 Information Flows

The number of lines in an activity chart can be reduced by grouping information elements into an **information-flow**, used to label a common flow line, see e.g. COMMANDS in the following figure, consists of SET_UP, EXECUTE, RESET.

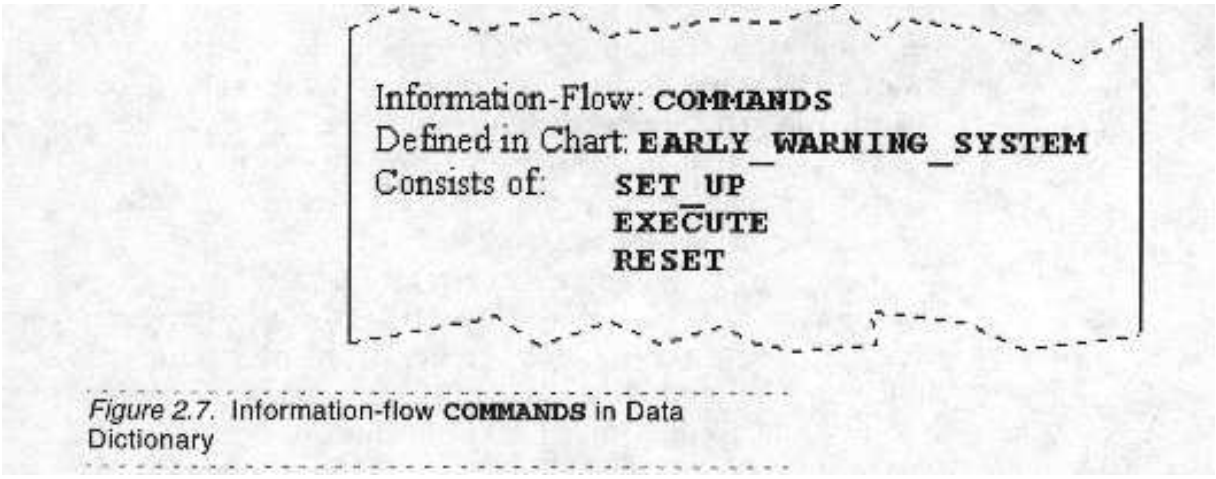



*Figure 2.7.* Information-flow **COMMANDS** in Data Dictionary

# 3.3.4 Data Stores

- There are no restrictions on the time that data reside on a flow line. Nevertheless it is often more natural to incorporate an explicit data store in the chart:

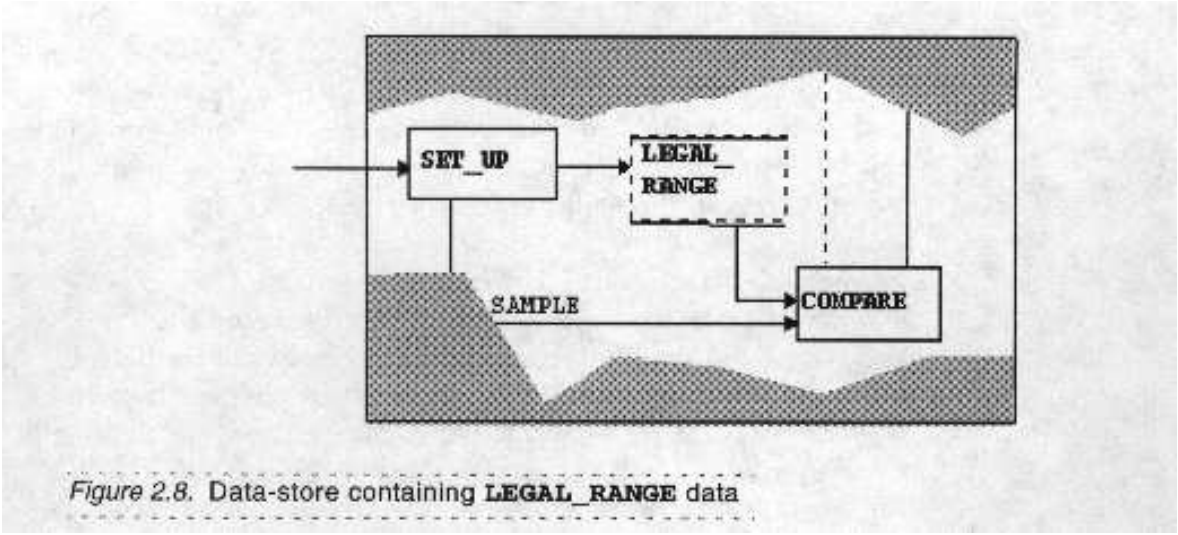



*Figure 2.8.* Data-store containing **LEGAL_RANGE** data

- A data item is defined in the Data Dictionary with the same name as the data store. Any structure given to a data item is inherited by the data store.

# 3.4 The Behavioral Functionality of Activities

- The behavior of subactivities of an activity chart is described by its **control activity**, whose function is to control their **sibling** activities (i.e., the other subactivities in the chart).

- A control activity may explicitly start and stop its sibling activities, i.e., EWS_CONTROL controls SET_UP, PROCESS_SIGNAL, and COMPARE:

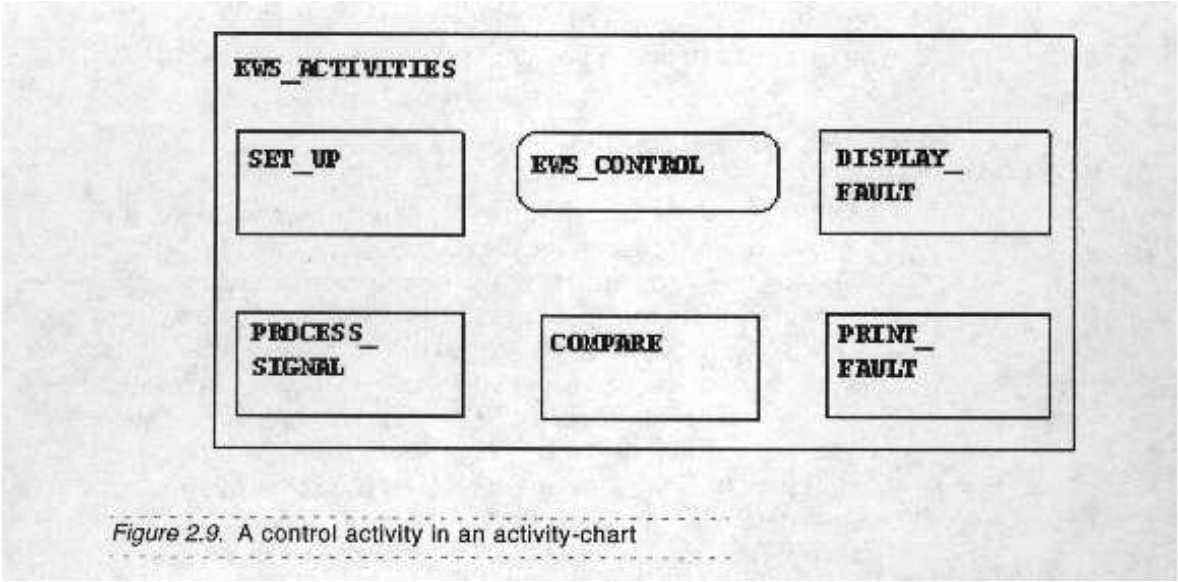
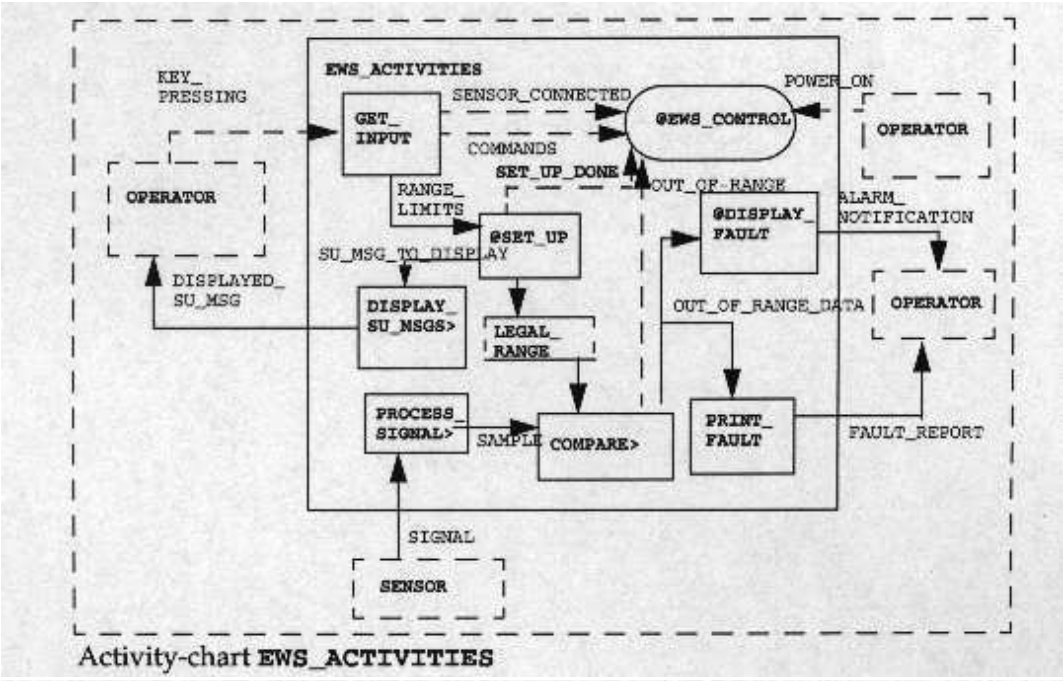


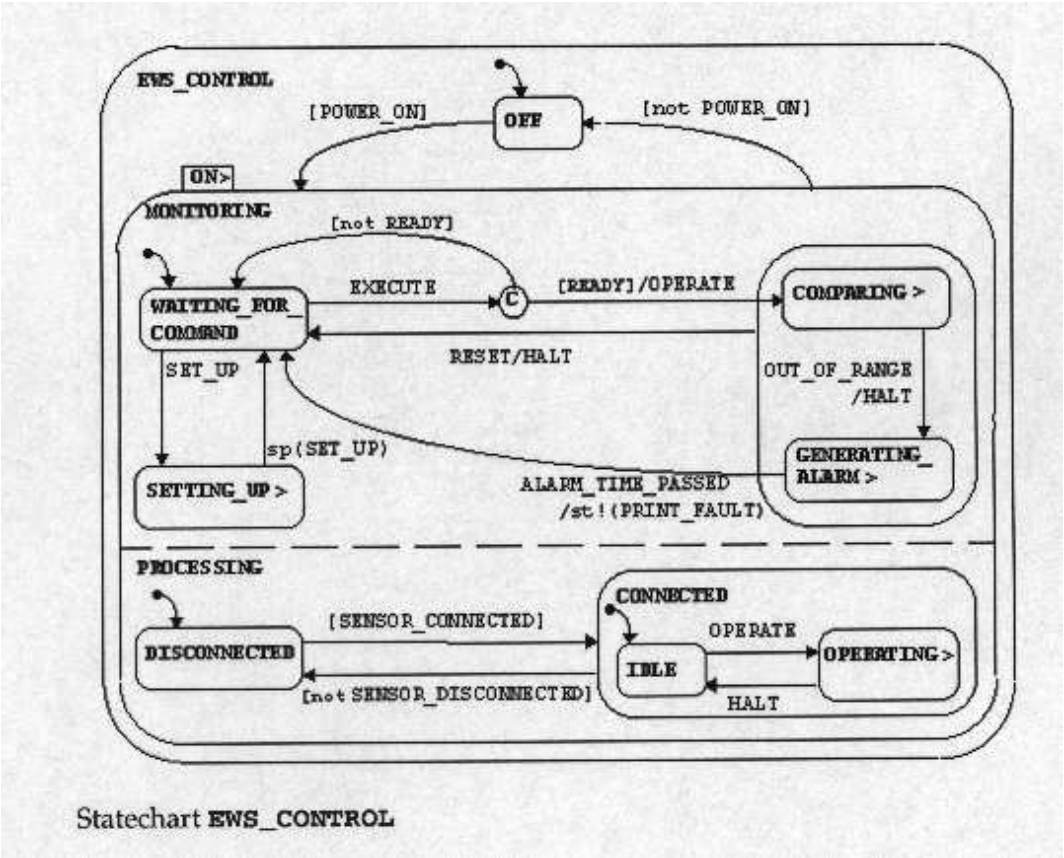

*Figure 2.9.* A control activity in an activity-chart

- Each activity may have **at most one** control activity.

- The control activity, depicted as a rectangle with rounded corners, cannot have subactivities. Rather its specification is that of a Statechart, see next slide.




Activity-chart **EWS_ACTIVITIES**




Statechart **EWS_CONTROL**

# 3.4.1 Activities in the Data Dictionary

- Every activity can be described more extensively in the Data Dictionary using **textual information**.
- **Basic activities** are described in the Data Dictionary by **executable textual descriptions**, specifying patterns of behavior. These patterns are:

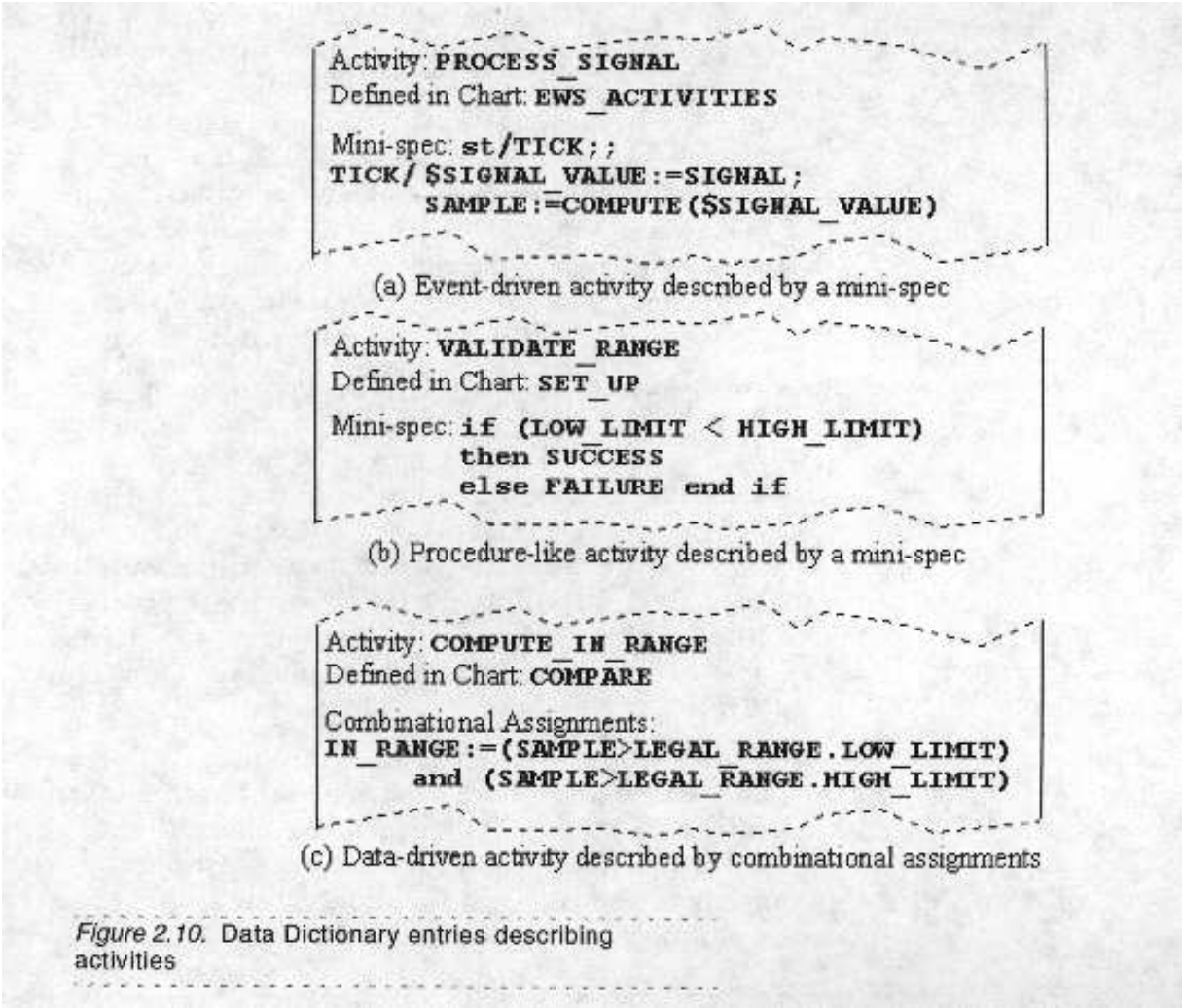



*Figure 2.10.* Data Dictionary entries describing activities

# 3.5 Connectors and Compound Flow-Lines

- The data flow lines leaving activity COMPARE in Figure 2.5 can be drawn with a **joint connector** as below:

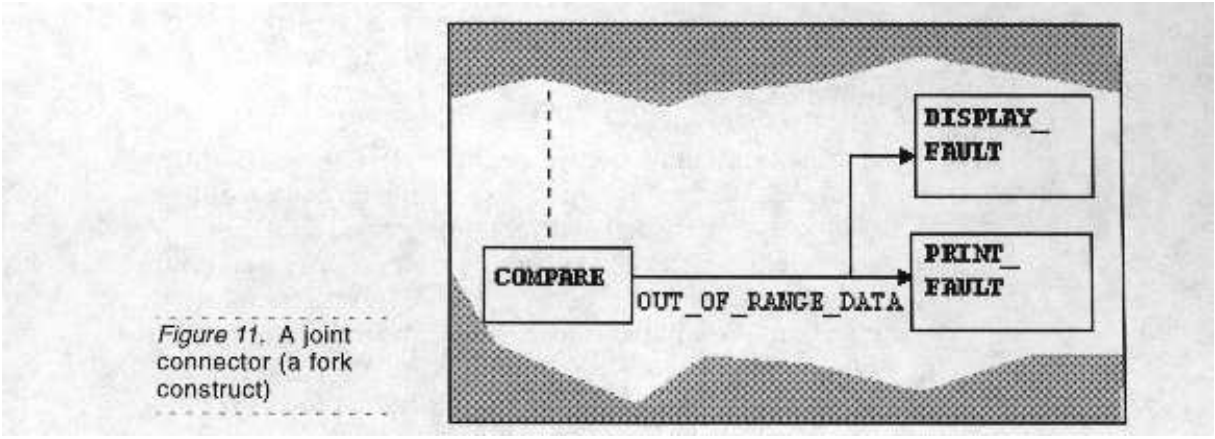



*Figure 11.* A joint connector (a fork construct)

# Junction connectors

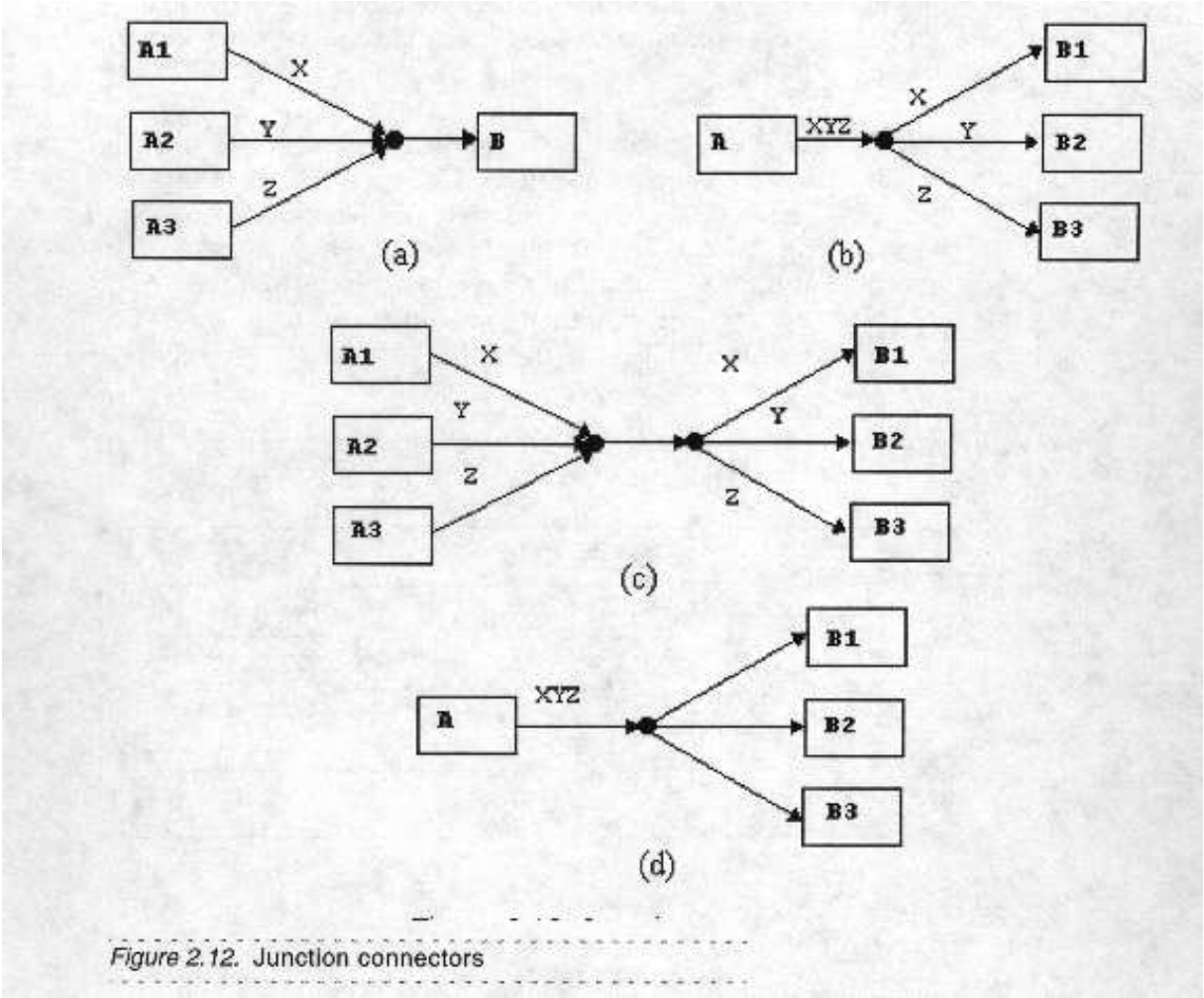



*Figure 2.12.* Junction connectors

# Diagram connectors

- **Diagram connectors** are used when the source of a flow line is far from its target:

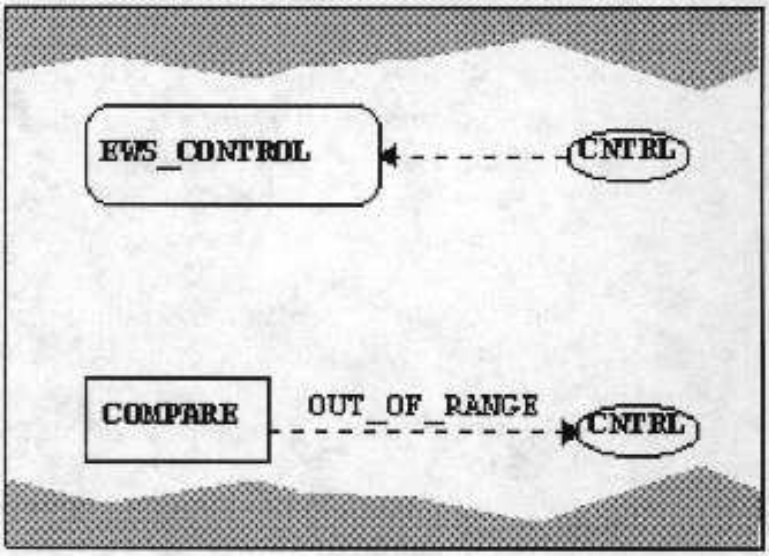



*Figure 2.14.* A diagram connector