Chapter

14

Hazard Analysis Models and Techniques

Before a wise man ventures into a pit, he lowers a ladder—so he can climb out.

—Rabbi Samuel Ha-Levi Ben Joseph Ibm Nagrela
Ben Mishle

Many different types of hazard analysis have been proposed and are in use. Some differ primarily in their names, whereas others truly have unique and important characteristics. One of the greatest problems in performing hazard analysis may be in selecting appropriate models and techniques that match the project's goals, tasks, and skills. Because the methods have different coverage and validity, several may be required during the life of the project. No one method is superior to all others for every objective or even applicable to all types of systems. Perhaps the most important fact to keep in mind is that very little validation of any of these techniques has been done, and so all results should be treated with appropriate skepticism. That does not mean that the techniques are not useful, only that they must be used carefully and combined with a large dose of engineering judgment and expertise.

The resources and time for any analysis are limited. Not all resources should be put into one single method or into one single phase of the analysis process. In planning the analysis and selecting appropriate procedures, consideration should be given to its purpose, who will use the results and what kind of information is expected, the seriousness of the potential hazards, the complexity of the system, the nature of the project and the uniqueness of its design and technology, the degree of automation, the types of hazards to be considered, and the role of humans

and computers in the system [299]. Only a few of the techniques described can handle software in any reasonable way.

In addition to the analysis techniques presented here, some of the accident models described in Chapter 10, along with appropriate analysis techniques, can be (and are) used in accident investigations and occasionally in predictive analyses.

In this chapter, each technique is first described in terms of its basic features and the life cycle phase to which it applies, and then it is evaluated. The final section of the chapter describes the small amount of experimental validation of techniques that has been carried out.

14.1 Checklists

Description

There is a tremendous amount of hard-earned experience in engineering, and checklists are one way to pass on this experience so that each project need not relearn the lessons of the past and start each hazard analysis from scratch. As the repository of mistakes made and lessons learned, checklists provide feedback to the engineering process.

When checklists are dynamically updated within an organization, they may become uniquely tailored to that organization's procedures and practices. They may also be derived from standards and codes of good engineering practice. Checklists are most useful in the design of well-understood systems, for which standard design features and knowledge have been developed over time.

Checklists are included as an analysis technique here because they guide thinking. In fact, many of the other analysis techniques incorporate some form of checklist in their procedures. Basic checklists are simply lists of hazards or specific design features. Others stimulate thought and enquiry with questions that are open ended rather than requiring only a "yes" or "no" answer or a check in a box. For example, instead of asking "Is the system protected against electromagnetic interference?" the list might instead ask "How is the system protected against EMI?"

Life-Cycle Phase

Checklists are commonly used in all life-cycle phases, and in fact are most useful when oriented toward a specific phase. For hazard identification, they provide information about known hazards or high-risk conditions, helping to make sure that hazards are not overlooked. For a design, they ensure conformance to existing codes and standards of practice. Design checklists often use a series of *what-if* questions like those in design reviews. During operations, checklists may be used for periodic audits or to ensure that steps in procedures are not forgotten

(like checklists used by pilots, for example). Information gained during the hazard analysis process for the project should be used to design operations checklists.

Evaluation

Checklists are an excellent way to pass on lessons learned, especially for hazard identification. For designers, they help to ensure good engineering design practices and compliance with standards, codes, and specifications; for reviewers, they help to verify that prohibited or bad practices have been avoided and that requirements have been satisfied.

On the negative side, checklists may encourage users to rely on them too much and thus to overlook items not on the list. Also, to be comprehensive, checklists may have to include a large number of questions, and, as experience reveals problems, more will be added. They can then become large and difficult to use, and users may be lulled into thinking that all issues that should be considered have been included. Checklists often induce false confidence—a belief that if everything is checked off, the system is safe. In addition, most do not allow relative ranking of hazards or include information about relative effectiveness of alternative safeguards [126].

Another problem arises when the lists are used without giving careful thought to the specific situation being considered. Ozog and Bendixen provide an example from the process industry: A checklist might reasonably require flame arrestors in vents from flammable liquid storage tanks, but if the vapors are susceptible to polymerization, venting directly to the atmosphere might be safer [250]. In this case, relying on the checklist without considering special circumstances might create a more hazardous situation.

While checklists may be useful, more sophisticated analyses for all but the simplest systems are essential to an effective safety program.

14.2 Hazard Indices

Description

Hazard indices measure loss potential due to fire, explosion, and chemical reactivity hazards in the process industries. They were originally developed primarily for insurance purposes and to aid in the selection of fire protection methods, but they can be useful in general hazard identification, in assessing hazard level for certain well-understood hazards, in the selection of hazard reduction design features for the hazards reflected in the index, and in auditing an existing plant.

The oldest and most widely used index was developed by the Dow Chemical Company. The *Dow Chemical Company Fire and Explosion Index Hazard Classification Guide* (usually abbreviated as the *Dow Index*) was first published in 1964 and was originally the basis for calculating a Fire and Explosion Index. Later, it was expanded to calculate the maximum probable property damage and

the maximum probable days outage. Any operation where a flammable, combustible, or reactive material is stored, handled, or processed can be evaluated with the *Dow Index*. Auxiliary plant, such as power generation equipment, office buildings, control rooms, or water systems, is not covered.

The *Dow Index* first requires dividing the plant into units, a unit being a part of a plant that can be readily and locally characterized as a separate entity. Generally, a unit consists of a segment of the overall process: In some cases, it may be a part that is separated by distance or by walls; in others, it may be an area in which a particular hazard exists [196, 126].

The Fire and Explosion Index indicates the fire and explosion hazard level of a particular unit. The calculation of this index uses a measure, called the *Material Factor* (MF), of the energy potential of the most hazardous material or materials in the unit in sufficient quantity to present a hazard. This measure is a number from 1 to 40 and is calculated on the basis of flammability and reactivity. For some properties, the MF can be found in a table; for others, it must be calculated (Lees [172] explains how). General and special hazards (including factors such as properties of the materials, quantities involved, the type of process and whether it is difficult to control, process conditions, and construction materials) are treated as penalties applied against the MF. A Toxicity Index can also be calculated to evaluate the exposure level of toxicity hazards.

Basically, these calculations combine a number of empirical hazard factors that reflect the properties of the materials being processed, the nature of the process, the spacing of equipment, and the judgment of the analyst about them [126]. The index is then used to determine the fire protection required. Basic fire protection design features, including minimum separation distances, are recommended in the *Dow Index*.

Attempts have been made to improve on this index or to come up with alternative indices, but most alternatives have not found widespread acceptance outside the organizations in which they were developed [249]. One that has been used in the chemical industry, called the *Mond Index*, was proposed in 1979. It expands the *Dow Index* to include additional factors related to the effects of toxic materials and layout features (such as spacing, access, height, and drainage) on the hazard level.

Evaluation

Hazard indices provide a quantitative indication of the potential for hazards associated with a given design. They work well in the process industry, where designs and equipment are standard and change little, but are less useful for systems where designs are unique and technology changes rapidly. Lowe and Solomon [196] claim that the *Dow Index* and others are particularly useful in the early stages of hazard assessment, since they require a minimum of process and design data and can graphically demonstrate which areas within the plant require more attention. The indices can also help to identify which of several competing process designs contain the fewest inherent hazards, and they provide information useful for site selection and plant layout [250].

The indices only consider a limited set of hazards, and even for these, they determine only hazard level. No attempt is made to define specific causal factors, which are necessary to develop hazard elimination or reduction measures beyond the standard equipment information provided in tables. Thus, the indices do not provide a complete picture and are useful primarily to supplement other hazard analysis methods.

14.3 Fault Tree Analysis

Description

Fault Tree Analysis (FTA) is widely used in the aerospace, electronics, and nuclear industries. It was originally developed in 1961 by H. A. Watson at Bell Telephone Laboratories to evaluate the Minuteman Launch Control System for an unauthorized (inadvertent) missile launch. Boolean logic methods had been used at Bell Labs for communications equipment, and these were adapted to FTA. Engineers and mathematicians at the Boeing Company developed the procedure further and became its foremost proponents.

FTA is primarily a means for analyzing causes of hazards, not identifying hazards. The top event in the tree must have been foreseen and thus identified first by other techniques. FTA uses Boolean logic to describe the combinations of individual faults that can constitute a hazardous event. Each level in the tree lists the more basic events that are necessary and sufficient to cause the problem shown in the level above it.

FTA is a top-down search method. Backward or forward search techniques are chronological orderings of events over time, but each level of the fault tree merely shows the same thing in more detail. The intermediate events (events between the top event and the leaf nodes in the tree) are pseudoevents (abstractions of real events)—they are simply combinations or sets of the basic or primary events and are usually removed during the formal analysis of the tree (Figure 14.1).

Once the tree is constructed, it can be written as a Boolean expression and simplified to show the specific combinations of identified basic events sufficient to cause the undesired top event. If a quantitative analysis is desired and feasible (the individual probabilities for all the basic events are known), the frequency of the top event can be calculated.

Fault Tree Analysis has four basic steps: (1) system definition, (2) fault tree construction, (3) qualitative analysis, and (4) quantitative analysis.

System Definition. This is often the most difficult part of the FTA task; it requires determining the top event, initial conditions, existing events, and impermissible events. The selection of top events is crucial, since the assessment of

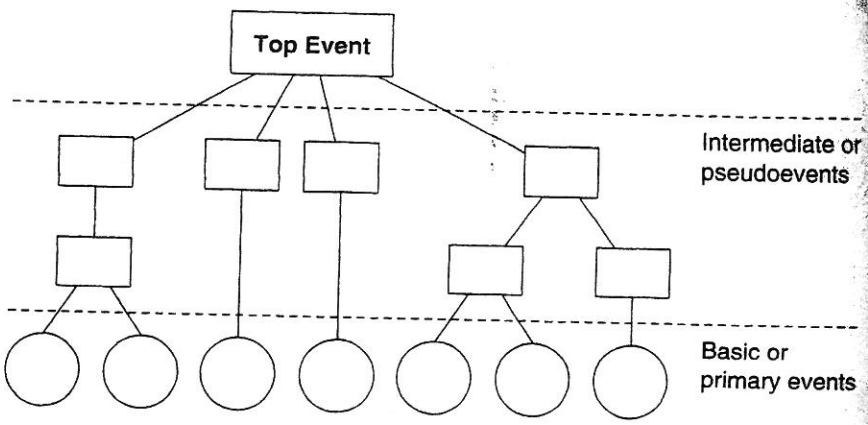


FIGURE 14.1
The leaf nodes of a fault tree represent the basic or primary events.

hazards in the system will not be comprehensive unless fault trees are drawn for all significant top events.

A thorough understanding and definition of the system and its interrelationships is essential for this step and all other steps in FTA. The analyst may use system functional diagrams, flow diagrams, logic diagrams, or other design representations, or may rely on his or her knowledge of the system. The physical system boundaries must be carefully defined.

For any component that has more than one possible state, the analyst must decide upon the system state (initial state) to be analyzed for the occurrence of the top event. If the top event is an inadvertent weapon release from an aircraft, for example, the events in the tree will be very different depending on whether the aircraft is on the ground, in flight and cruising to target, or over the target but not in proper position for the release [296]. Similarly, the fault tree for the collision of two automobiles will depend upon traffic speed and density.

Fault Tree Construction. Once the system has been defined, the next step is fault tree construction. Briefly, the analyst first assumes a particular system state and a top event and then writes down the causal events related to the top event and the logical relations between them, using logic symbols to describe the relations. Figure 14.2 shows the symbols used for fault trees, of which the most frequently used are AND and OR gates. The output of an AND gate exists only if all the inputs exist (it represents combinations of events); the output of an OR gate exists provided at least one of the inputs exists (it shows single-input events that can cause the output event). The input events to an OR gate do not cause the event above the gate, but are simply re-expressions of the output event. In contrast, the

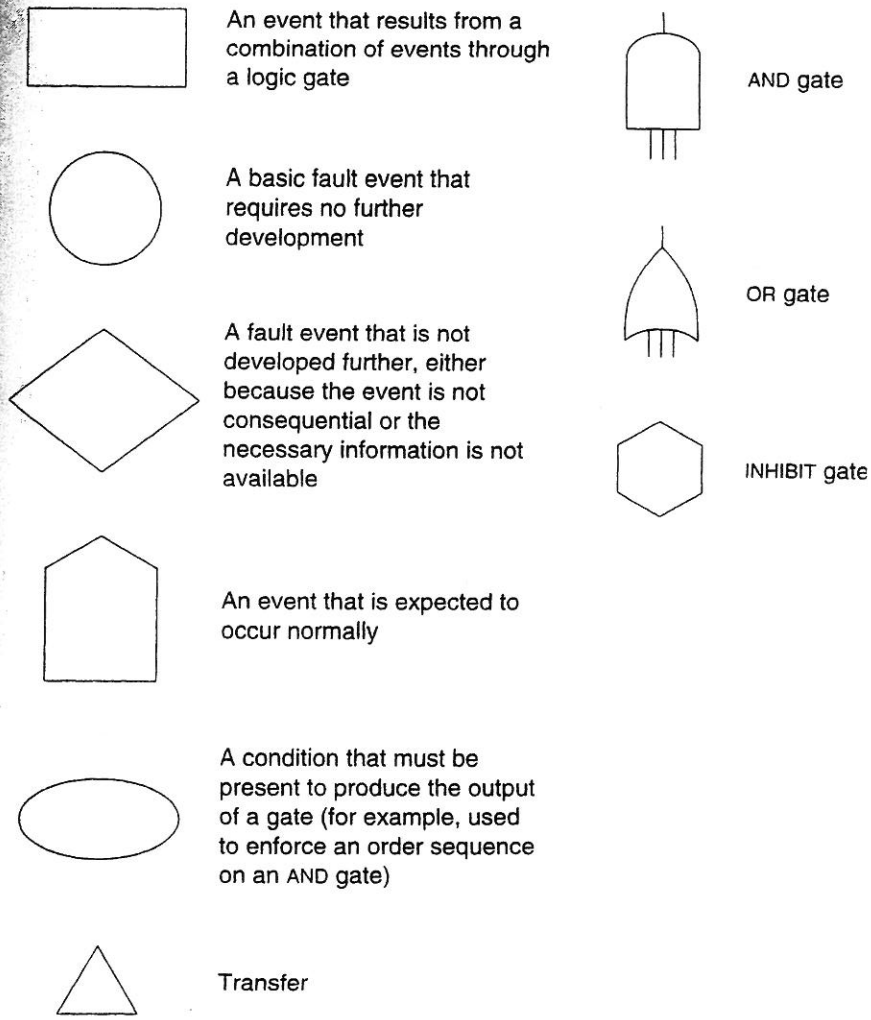


FIGURE 14.2
Fault tree symbols.

events attached to the AND gate are the causes of the event above the gate [344]. This causal relationship is what differentiates an AND gate from an OR gate.

Inhibit (NOT) gates can also be used, but are less common. They may be needed in a situation where there are two flows, X and Y, and the top event occurs if there is either no X flow or no Y flow. The simple OR gate is inclusive—it states that the top event occurs if there is a failure of X flow, Y flow, or both. If the goal is to specify that the top event does not occur if there is a failure of *both* X and Y flows (exclusive OR), then the simple OR gate will not suffice and an INHIBIT or other type of gate is needed.

The relationships between the events shown in the fault tree are just standard logical relations and therefore can be expressed using any of the alternative forms of Boolean algebra or truth tables. The tree format, however, seems to have advantages in terms of readability.

The process continues, with each level of the tree considered in turn until basic or primary events are reached. These are completely arbitrary, and the analyst must determine the stopping rule for the analysis, or, in other words, the resolution limit of the analysis. The events considered to be basic in the analysis will depend on its purpose, scope (a first estimate or a fully detailed analysis), and intended users; the available knowledge about the causes of events; and the availability of statistical data if a quantitative analysis is desired. Figure 14.3 is an example of a fault tree.

Qualitative Analysis. After the tree is constructed, qualitative analysis can begin. The purpose, basically, is to reduce the tree to a logically equivalent form showing the specific combinations (intersections) of basic events sufficient to cause the top event. In essence, the intermediate pseudoevents are removed and only relationships between the top event and the primary events are described. These are called *cut sets*. The goal of the analysis is to find the *minimal cut sets*, which represent the basic events that will cause the top event and which cannot be reduced in number—that is, a cut set that does not contain another cut set. Cut sets are defined such that if even one event in the cut set does not occur, the top event will not take place.

The minimal cut set representation as a tree corresponds to one OR gate with all the minimal cut sets as descendants. The same primary events usually will occur in more than one of the minimal cut sets; thus, the minimal cut sets are generally not independent of each other. A medium-sized fault tree can have millions of minimal cut sets, so computer programs have been developed to calculate them. The procedures for reducing the tree to a logically equivalent form are beyond the scope of this book; the interested reader is referred to one of the many books on this subject. In general, the procedures employ Boolean algebra or numerical techniques, for example using the logical structure of the tree as a model for trial and error testing of the effects of selected combinations of primary events.

Minimal cut sets provide information that helps identify weaknesses in the system. For example, they determine the importance or ranking of each event with respect to the top event. A number of measures of importance have been

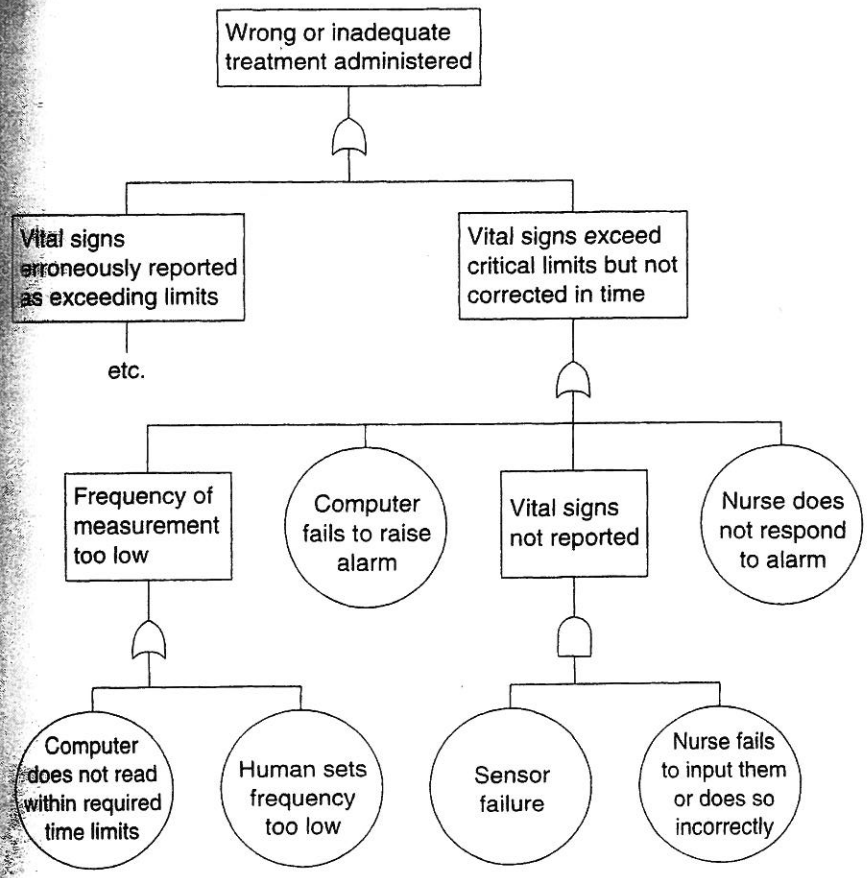


FIGURE 14.3
Portion of a fault tree for a patient monitoring system.

defined; some rely solely on structural considerations, while others require probabilistic information [172].

Quantitative Analysis. The probability of the output of a logical gate is equal to the probability of the corresponding function of the input events—both characterize the same event. Quantitative analysis of fault trees uses the minimal cut sets to calculate the probability of occurrence of the top event from the probability of occurrence of the basic events. The probability of the top event will be the sum of the probabilities of all the cut sets if they are all statistically independent (the same event is not present in two or more cut sets). If there is any replication of

events in any cut set, independence is compromised and the replication must be taken into account in any quantitative analysis. The probabilities of each cut set are determined by multiplying together the probability of the basic events.

According to Ozog and Bendixen [250], a common mistake in quantifying fault trees is multiplying two or more frequencies together, yielding meaningless results. To help avoid this mistake, they have changed the tree symbols to clarify which events are frequencies and which are probabilities.

If the probabilities of the basic events are given by a probability density function (the range of probabilities over which the event can occur) rather than by a point probability value, then the probability of the top event also must be expressed as a density function. Monte Carlo simulation can be used to determine these functions [126].

Automatic Synthesis. Several procedures for automatic synthesis of fault trees have been proposed, but these work only for systems consisting purely of hardware elements. Basically, a model of the hardware, such as a circuit diagram, is used to generate the tree [7, 10, 172]. Taylor's technique, which is typical, takes the components of the hardware model and describes them as transfer statements [335]. Each statement describes how an output event from the component can result from the combination of an internal change in the component and an input event. Such statements can also describe how the component state changes in response to input events. In general, the transfer statement will be conditional on the previous component state. Together, the transfer statements form the transfer function for the component.

Both the normal and failure properties of the component are described, and each transfer statement is represented as a small fragment of a fault tree or mini-fault tree. The synthesis process consists of building the fault tree by matching the inputs and outputs of the mini-fault trees. The same type of analysis can be done using state-machine models (see Section 14.12).

Software FTA. Fault tree analysis can be applied to software, as described in Chapter 18. With software, the analysis is used for verification, as the code must already have been written to generate the trees. FTA might be applied to software design representations to locate problems early, but the design specification would have to be very detailed in its description of the software logic. Once the code is generated, not only is building the tree difficult, but so is changing the software in any significant way. Software fault trees can be partially generated automatically; however, if loops are used in the code (which is true for virtually all software), the tree generation requires human assistance.

Probabilistic analysis is not applicable when software logic is described by fault trees; assigning a probability to a software statement is basically meaningless. In addition, if design errors are found in the tree through this process, they should be fixed rather than left in the code and assigned a probability.

Life-Cycle Phase

Although generic fault trees can be constructed before the details of design and construction are known, they are of limited usefulness. To be most effective, FTA requires a completed system design and a thorough understanding of the system and its behavior in all operating modes. Information is usually too incomplete to perform detailed fault tree analysis at the preliminary design stage [5], although a few features of alternative designs may be compared. In addition, fault trees can be used early in the design process to identify where interlocks are required and will be most effective, but they are not the most efficient model for this process [42].

FTA may also be applied to completed or existing systems to prove that the system is safe. The quantitative fault tree procedures are most useful for this purpose.

Evaluation

Although FTA was originally developed to calculate quantitative probabilities, it is more commonly used qualitatively. Simply developing the tree, without analyzing it, forces system-level examination beyond the context of a single component or subsystem. The graphical format provides a pictorial display of the relationships between events and helps both in understanding the system and in detecting problems or omissions in the analysis. Problems are also found because the analyst has to think about the system in great detail during tree construction.

Fault trees can help the analyst identify scenarios leading to hazards and can suggest possibilities for hazard elimination or control even before any analysis is performed on the tree. When software is part of the system, drawing the fault tree down to the software interface with other system components will identify safety-critical interfaces and potentially hazardous software behavior.

Knowing the minimum cut sets for a particular fault tree can provide valuable insight into potential weak points of a complex system, even when it is not possible to calculate the probability that either a particular cut set or the top event will occur. Lewis describes three useful qualitative considerations [189]. First, the ranking of minimal cut sets by the number of primary events required allows emphasizing the elimination of cut sets corresponding to small numbers of events. Single-point failures (where the occurrence of a single event could cause a hazard) can be uncovered and eliminated (they appear as a cut set containing a single event).

Second, events or components that appear in several minimum cut sets for a particular top event are likely to have an important effect on the occurrence of that event. In addition, if events or components appear only in minimum cut sets requiring several independent events, their importance with respect to the top event is likely to be small. The result of assessing importance in this way is a prioritized list of events that should be considered in reducing risk.

The independence of the events must be determined, and this determination

is the third use for qualitative analysis—to focus common-cause failure analysis on particular cut sets and events. The events can be examined for susceptibility to common influencing factors such as weather or temperature extremes, vibration, corrosion, and environmental conditions such as dust or humidity. Even so, potential common-cause failures are not always obvious from the FTA process unless the analyst is very experienced and knowledgeable.

Most methods to handle common-cause failures in fault trees are qualitative only, but identification is more important than quantification anyway. Once a common-cause failure mode is identified, usually it can be eliminated completely—if there is enough information to measure it, there is usually enough information to eliminate it. A different type of common-cause failure, which occurs by fault propagation (domino effects), is also possible, but there appears to be no way of treating this type of failure in fault trees. Common-cause failures are important because, in very high reliability systems, they can become a dominant factor in system reliability and in accidents [172].

The extra work of a quantitative analysis may be cost effective when there are very subtle differences between several alternative designs [51] and when the causal factors involved have well established and accurate probabilities. The impact of the alternatives on the top-event frequency is calculated to determine the impact of the design decisions and the safety or reliability tradeoffs involved.

Fault Tree Analysis has several limitations. The most useful fault trees can be constructed only after the product has been designed; they require detailed knowledge of the design, construction, and operation of the system. A good safety program, however, requires concentration on the early stages of the system life cycle. Generic fault trees can be built early, but they may provide only information that is well known and already part of the project standards and design criteria. Hammer says that it may be better to spend time ensuring that the design criteria have been incorporated than building fault trees [108]. Childs notes that sometimes fault tree analysis finds only what is intuitively obvious [51]. One use for FTA in the design stage is to trace system hazards to individual components—such as software—in order to identify hazardous component behavior.

Fault tree analysis shows cause and effect relationships but little more. Additional analysis and information is usually required for an effective safety program. Moreover, reliability analysts usually concentrate only on failure events in fault trees whereas hazard analysis (as opposed to reliability analysis using fault trees) requires a broader scope. Thus, the use of fault trees by reliability analysts may differ from their use by system safety analysts. Applications of fault tree analysis that focus primarily on failures are essentially just reliability analyses.

A fault tree, like any other model, is a simplified representation of a generally very complex process: Its relative simplicity can be deceptive [172]. Much of the work on FTA is concerned with correcting the oversimplifications, but the problems might be better overcome by using different types of models and analyses to handle these factors directly rather than trying to forcefit everything into one (perhaps inappropriate) analysis framework. For example, the [illegible]

is particularly suited to discrete events, such as a valve opening or closing, but time- and rate-dependent events, such as changes in critical process variables, degrees of failure (partial failure), and dynamic behavior are not so easily represented [126].

Simple AND and OR gates do not convey any notion of time ordering or time delay; the fault tree is a snapshot of the state of the system at one point in time. In some cases, time spans or chronological ordering of events may need to be specified. Other types of gates, such as DELAY and INHIBIT, allow some treatment of time in fault trees and in the tree reduction process [255]. They complicate the evaluation of the tree, however, and somewhat negate one important advantage of FTA—the ease with which the trees can be read and understood and thus reviewed by experts and used by designers. If chronology is important, using a model and analysis technique that involves backward or forward search may be more appropriate than forcefitting this into a hierarchical, top-down modeling technique.

Transitions between states are not represented in fault trees, which deal best with binary states: Partial failures and multiple failures can cause difficulties [62]. Because system states rather than sequences of states are shown, fault trees are used less often in studies of batch systems and plants (where sequence is important) than in continuous systems. Nonaction or static systems (such as pressure vessels) are also difficult to handle, since their state depends primarily on environmental events or event combinations rather than on the component state itself [5].

Problems also occur in the analysis of *phased-mission* systems, which pass through more than one phase of operation [172]. Typically in these systems, the same equipment is used at different times and in different configurations for different tasks, and thus a separate fault tree is needed for each phase. While it is possible to think of this type of system as essentially an OR gate under the top event, where the inputs to the OR gate represent the different phases of the mission, the standard OR will not suffice because the inputs are separated in time. Although phased-mission systems can be handled by constructing several fault trees, problems can occur at the phase boundaries that are not easily resolved [172].

Additional criticisms of fault tree analysis relate to its quantitative aspects. As mentioned, common-cause failures cause problems and can lead to orders-of-magnitude errors in the calculated failure probability [217].

As with any technique that tries to quantify factors in complex systems probabilistically, data may not be available for the most important factors, such as operator work conditions, the management system, design errors, human errors of various kinds, and nonrandom failures and events. Either these factors are left out because they cannot be quantified, or probabilities are assigned that are unrealistic or have very large uncertainties. Combining reliabilities of parts containing five or six significant figures with human error probabilities having significant uncertainties does not produce very useful conclusions. Misleading results can also be obtained by using data that is not applicable because conditions are not similar

to those under which the data was obtained or by averaging widely different data (one can drown in a lake with an average depth of six inches) [156].

Actually, most errors in hazard analysis are not due to errors in the data but to the failure to foresee all the ways in which the hazard could occur. According to Kletz, "time is usually better spent looking for all the sources of hazard than in quantifying with ever greater precision those we have already found" [156]. MacKenzie notes that because it is not possible to identify all accident sequences, the absolute values of the calculated risks have large uncertainties. In the space program, where quantitative Fault Tree Analysis (and Failure Modes and Effects Analysis) was used extensively, almost 35 percent of the actual in-flight malfunctions had not been identified by the technique as "credible" [205], as noted earlier.

14.4 Management Oversight and Risk Tree Analysis

Description

Management Oversight and Risk Tree analysis (MORT), developed by Johnson in the 1970s for the U.S. Nuclear Regulatory Agency, was discussed in Chapter 10 as an accident model. It can also be used as an accident investigation or hazard analysis technique. The underlying accident model assumes that accidents are caused by uncontrolled energy released by mishandled changes in the system.

Basically, MORT is a standard fault tree augmented by an analysis of managerial functions, human behavior, and environmental factors. Its aim is to identify problems, defects, and oversights that create hazards or prevent their early identification by poor planning, inadequate operational checks, or limited information exchange within the organization.

The method uses an extensive checklist of 1,500 basic events or factors that facilitates finding those safety problems included in the list. Figure 10.10 shows an example of MORT [140].

Evaluation

MORT has the advantages and disadvantages of any checklist-based analysis. An advantage over most of the other analysis methods described in this chapter is its consideration of factors related to the organization, information system, management practices, and principles and goals of the enterprise. Relatively little emphasis, in practice, has been placed on management and human factors (aside from trying to measure human errors so numbers can be attached to basic events in fault trees) compared to the emphasis on the reliability analysis of engineered systems [322].

Suokas suggests that, in his experience, MORT analysis yields detailed information useful in the planning and coordination of activities involving several departments, a more precise definition of important tasks and responsibilities, the development of planning and operating procedures, and training in professional

abilities and safety matters [324]. MORT is not used very often, perhaps, as suggested by Suokas and Kakko, because of its complexity.

14.5 Event Tree Analysis

Description

FTA is the most widely used method for the quantification of system failures, but it becomes very difficult to apply in complicated systems. WASH-1400 was a complex, probabilistic risk assessment of nuclear power plants in the early 1970s. The study team (see Chapter 8.5.2) first attempted to draw a fault tree for nuclear reactors starting with the top event, "accidental release of radioactivity," but they gave up when this led to a hopelessly complicated fault tree [281]. Instead, they adapted the general decision tree formalism, widely used for business and economic analysis, to break up the problem into smaller parts to which FTA could be applied.

This decision tree technique, called Event Tree Analysis when used in this way, uses forward search to identify the various possible outcomes of a given initiating event, such as the rupture of a pipe, by determining all sequences of events that could follow it. The initiating event might be a failure of a system component or some event external to the system. The problem in any forward search, of course, is knowing where to start. In nuclear power plants (the principle application of this method), an accident is defined as any failure of the operating system that might result in the release of radioactivity. Thus, the starting point for listing the initiating events to be considered is the potential failures previously identified and defined by many years of safety analysis and by the licensing process for commercial nuclear power plants.

The states in the forward search are determined by the success or failure of other components or pieces of equipment. In nuclear power or other applications, where the stress is on protection systems, all the protection systems that can be used after the accident are first defined and then structured as headings for the event tree. The engineered protection functions are listed left to right in chronological order after the initiating event. The ordering of the headings on the event tree is important.

The event tree is then drawn from left to right, with branches under each heading corresponding to two alternatives: (1) successful performance of the protection system (the upper branch) and (2) failure of the protection system (the lower branch). After the tree is drawn, paths through it can be traced by choosing a branch under each successive heading, where each path corresponds to an accident sequence.

An example from the WASH-1400 report is shown in Figure 14.4. Here, the headings are pipe break, electric power, emergency core cooling system (ECCS), fission product removal, and containment integrity. Each of these systems is assumed either to succeed or fail in its specified function. The expression at the

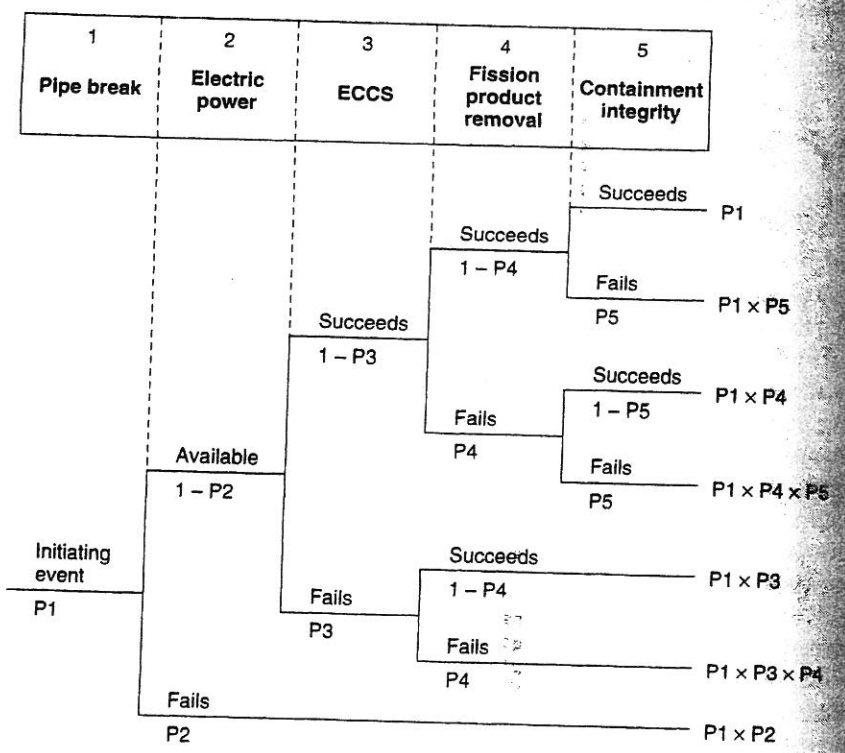


FIGURE 14.4
A reduced event tree for a loss of coolant accident. (Source: *Reactor Safety Study*, U.S. Nuclear Regulatory Commission, WASH-1400, NUREG 75/014, October 1974.)

right of each path is the probability for that path. Because the probability of failure is assumed to be very small, the probability of success is always close to 1. Therefore, the probability associated with the upper (success) branches of the tree is assumed to be 1.

Event trees tend to get quite large. They are reduced by eliminating sequences whose functional and operational relationships are illogical or meaningless. Since the system states on a given branch of the tree are conditional on the previous states having already occurred, another way to prune an event tree is to eliminate all branches that have a zero conditional probability for at least one event.

A path's total probability is found by multiplying together the probabilities at the various branches of the path, and the total risk of an accident is found by combining the path probabilities for all paths leading to an accident. The initial

event is expressed as a frequency (events per year), while the other, secondary events are probabilities (failures per demand). The probabilities for protection system failures (secondary events) are often determined using fault trees.

Event Tree Analysis is usually applied using a binary state system, as explained above, where each branch of the tree has one failure state and one success state. If a greater number of discrete states are defined for each branch (for example, if partial failures are included), then a branch must be included for each state. A problem is the possible explosion in the number of paths—for a sequence of N events, there will be 2^N branches of a binary tree. The number can be reduced by eliminating impossible branches, as described, but a large number of paths can still result.

Usually, a finite number of branches is defined at each node, but there is no conceptual problem with introducing a continuous random variable in an event tree [255]. Graphically, the spectrum of possible values of the continuous variable is represented by a fan originating at the event node. The analysis, in this case, uses a continuous conditional probability density and provides continuous joint distributions. In practice, a discrete variable may be more convenient, but in theory a continuous variable could be used [255].

Timing issues can cause problems in event tree construction. In some cases, failure logic changes depending on when the events take place. This happens, for example, in the operation of emergency core cooling systems in nuclear power plants [217]. As with fault trees, phased-mission analysis techniques are then needed to model the system changes during the accident sequence, even though the protection system does not change.

Another consideration is possible dependencies between the various probabilities arising from common-cause failures. In the nuclear reactor example, the value of the probability of ECCS function failure may depend in some way upon the conditions created by the pipe break itself. Such dependencies must be identified and assessed in the analysis, or the results can be distorted [5].

Life-Cycle Stage

Like Fault Tree Analysis, ETA is appropriate only after most of the design is complete. Thus, it has been used primarily to evaluate existing plants or designs. Note that by definition, and by the use of protection systems as the headings for the event tree, a decision is made in advance that the solution to the problem of safety will be to use protection systems. ETA does not require these headings, but it is difficult to determine which events to use for the headings otherwise. A general forward analysis of this type that did not drastically limit the events to be considered would be potentially enormous.

Evaluation

Fault trees lay out relationships between events: They are snapshots of the system state. Event trees, in contrast, display relationships between juxtaposed events

(sequences of events) linked by conditional probabilities. As a result, at least in theory, event trees are better at handling notions of continuity (logical, temporal, and physical), while fault trees are more powerful in identifying and simplifying event scenarios. Event trees allow the direct introduction of time factors and continuous random variables, but they are more than fault trees because of the potentially large number of branches. Combinations of events can be more concisely represented in fault trees using logical functions. Figure 14.5 shows the same event represented by a fault tree and an event tree. The accident modeled is described in Section 4.1.1; here, a computer has been added to the original design. Notice that a top-down search model like a fault tree loses the information about the ordering of relief valve operation (although it could be added by adding more complex types of tree structures), while the forward-search event tree model does not include detailed evaluation of the individual events.

Event trees are useful within the scope for which they were devised—probabilistically evaluating the effects of protection system functioning and failure in an accident sequence, particularly when events can be ordered in time. They are practical when the chronology of events is stable and the events are independent of each other [173].

Event trees can be helpful in (1) identifying the protection system features that contribute most to the probability of an accident, so that steps can be taken to reduce their failure probability; (2) identifying top events for subsequent fault tree analysis; and (3) displaying various accident scenarios that may result from a single initiating event.

Like all the analysis techniques discussed in this chapter, event trees have many limitations. For one, they can become exceedingly complex, especially when a number of time-ordered system interactions are involved [58]. A complete risk analysis of a complex plant, using a combination of event trees and fault trees, will require many person-years of effort along with a number of simplifying assumptions. In addition, the use of FTA to determine the probabilities for many of the event tree branches may make it more difficult to identify common causes of failures [250].

A separate tree is required for each initiating event, making it difficult to represent interactions between event states in the separate trees or to consider the effects of multiple initiating events. In addition, while the event tree enumerates all possible combinations of component states related to an initiating event, it offers no help in determining whether the component failure combinations (paths) lead to system failure. Either the system is simple enough and the mapping can be done for each failure scenario without more formal analysis, or the system is more complex and fault trees have to be used to identify the failure modes [255].

The usefulness of event trees depends on being able to define the set of initiating events that will produce all the important accident sequences. For nuclear power plants, where all the risk is associated with one hazard (serious overheating of the fuel) and designs are fairly standard, defining this set of hazards may be easier than for other systems. Whether it can be done completely, even for nuclear power plants, is still undetermined. Similarly, defining the functions across

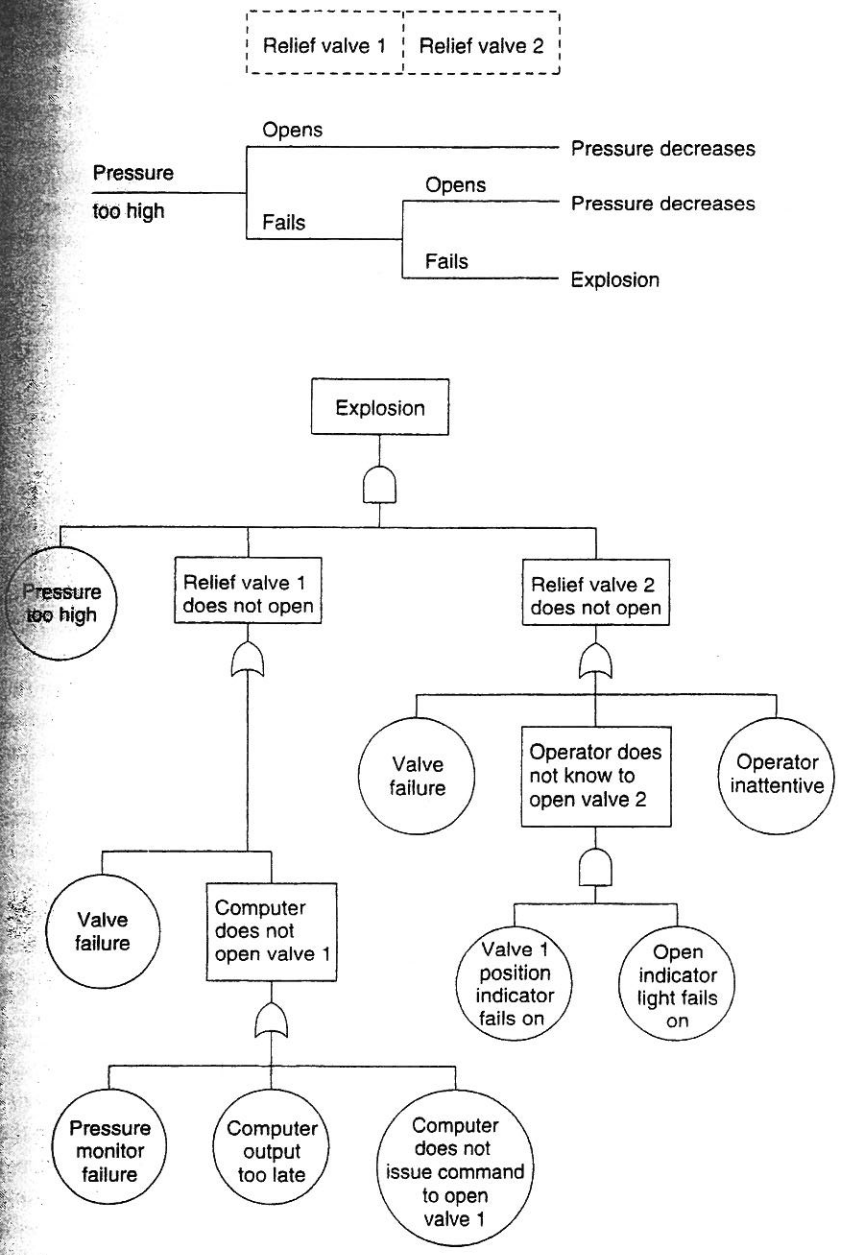


FIGURE 14.5
A fault tree and event tree comparison.

the top of the event tree and their order is difficult. Again, in nuclear power plants where responsibility for safety is vested in a specific set of protection systems, the events to use are more obvious than in other systems, although the problem of ordering is still there. Order is important when the performance of one system affects the performance of another. To solve the ordering problem, the analyst needs a detailed understanding of all plant systems, how they operate, and how they interact with one another [281]. As in most of these analysis techniques, building the model requires the interaction of analysts with different areas of expertise.

Finally, as with fault trees, continuous, nonaction systems such as dams are not appropriate for event tree analysis.

14.6 Cause–Consequence Analysis

Description

Cause–Consequence Analysis (CCA) is a relatively new technique developed by Nielson in the 1970s that combines several search modes [240]. CCA starts with a *critical event* and determines the causes of the event (using top-down or backward search) and the consequences that could result from it (forward search). The cause–consequence diagram shows both time dependency and causal relationships among events.

The procedure starts with the selection of a critical event, which is followed by a search for factors that constitute the critical event and by a propagation of the potential effects of the event. Finally, the interrelationships of the factors are described by a graphical model (see Figure 14.6).

Several cause charts may be attached to a consequence chart. The cause charts describe the alternative prior event sequences that can lead to the critical event and the conditions under which these sequences can occur. According to Nielson, the initiating events should be traced back to spontaneous events covered by statistical data [240]. Other cause charts attached to the consequence chart may be conventional fault trees, which show the combination of conditions under which a certain event sequence in the consequence chart can take place.

A table of symbols used in CCA is shown in Figure 14.7. The event and condition symbols describe the type of event or condition. The logic symbols include *gates* to describe the relations between cause events, and *vertices* to describe the relations between consequences. Standard AND and OR relations are the main logic gates and vertices. Another useful vertex is EITHER/OR, or the decision box, which specifies the effect of an event or condition on the paths the system takes. If the NO output from the decision box is the result of an abnormal condition, then the fault tree for this condition is derived. Thus, fault trees are used in the diagram not only for the critical event but also for abnormal conditions [172]

Taylor has shown how cause–consequence diagrams can be formalized to provide a semi-automatic analysis method [333]. The plant is represented by a

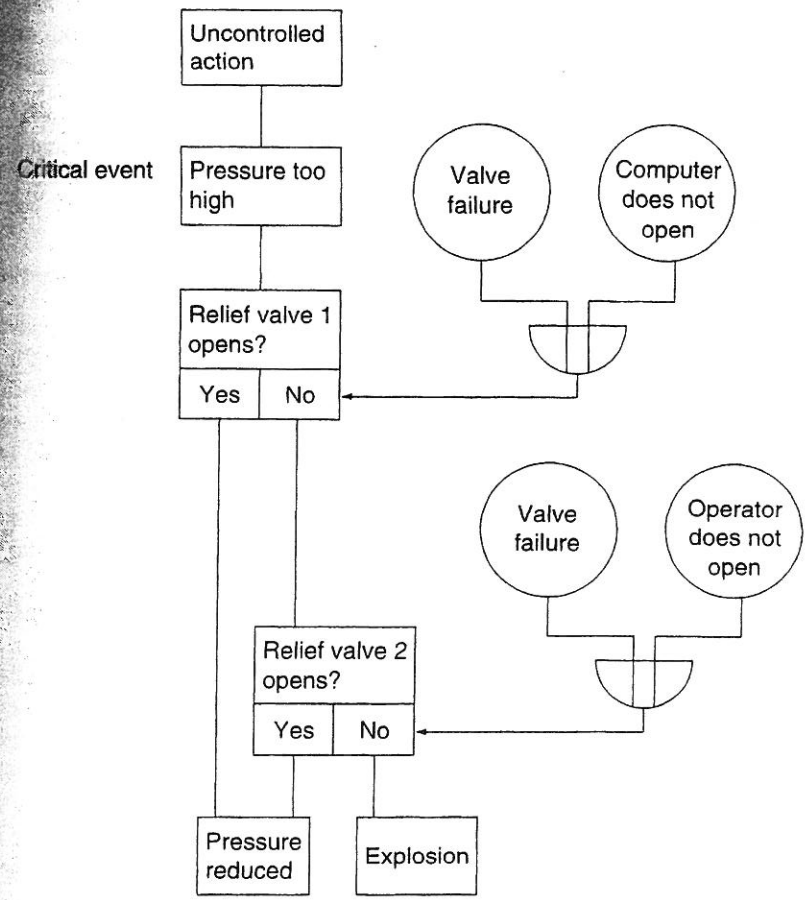


FIGURE 14.6
A cause–consequence diagram.

block diagram, where the arcs represent causal links. The blocks are described by arithmetic or transfer functions, as described earlier. A *condition* is a predicate that restricts the possible states of a system—usually by restricting the range of values of a single system variable—while an *event* is described by a pair of pre- and post-conditions (predicates true before and after the event, respectively). Event sequences can be traced through the block diagram to deduce the next event at each block.

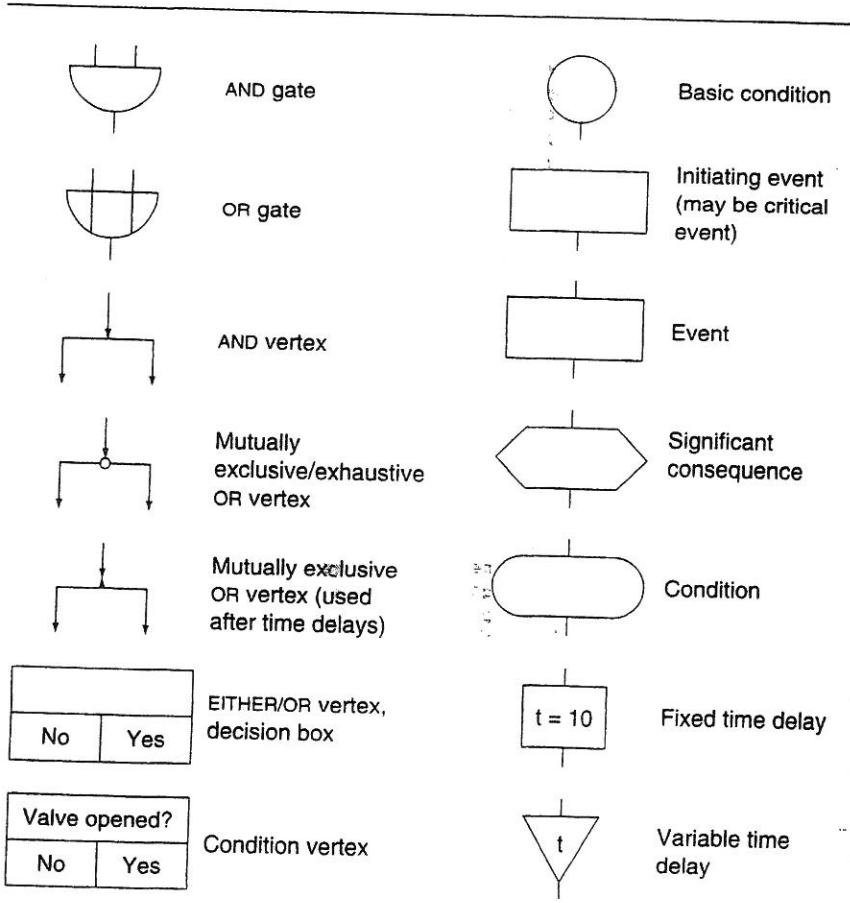


FIGURE 14.7
Cause–consequence diagram symbols.

Evaluation

Compared to fault trees, CCA shows the sequence of events explicitly, which makes the diagrams especially useful in studying startup, shutdown, and other sequential control problems. A systematic technique exists for constructing the diagrams (and also fault trees) from a block or wiring diagram of the plant.

Cause–consequence diagrams have the advantage over event trees of allowing the representation of time delays, alternative consequence paths, and combinations of events. They also take account of external conditions and the temporal ordering of events, where these factors are significant. Like the other techniques, CCA may be used for quantitative assessment.

On the negative side, the diagrams can become unwieldy, separate diagrams are required for each initiating event, and outcomes are related only to the cause being analyzed, although they could be caused by other initiating events [62]. CCA seems to be used more in Europe than in the United States.

14.7 Hazards and Operability Analysis

Description

Hazards and Operability Analysis (HAZOP) was developed by Imperial Chemical Industries in England in the early 1960s and later improved upon and published by the Chemical Industries Association in London. According to Ozog, about half the chemical process industry now uses HAZOP for all new facilities [249]. As the name suggests, the technique focuses not only on safety but also on efficient operations. Although it is usually applied to fixed plants, Kletz describes an application to tank trucks, in which several previously undetected hazards were identified and eliminated or controlled [161].

HAZOP is based on a systems theory model of accidents that assumes accidents are caused by deviations from the design or operating intentions—such as no flow or backward flow when there should be a forward flow. Basically, the technique encourages creative thinking about all the possible ways in which hazards or operating problems might arise. To reduce the chance that anything is forgotten, HAZOP is performed systematically, considering each process unit in the plant (such as pipelines, tanks, and reactors) and each hazard in turn. Questions are generated about the design by a small team of experts. Although prompted by a list of guidewords, the questions arise creatively out of the interaction of the team members [161].

HAZOP is a qualitative technique whose purpose is to identify all possible deviations from the design's expected operation and all hazards associated with these deviations. In comparison with hazard identification techniques like checklists, HAZOP is able to elicit hazards in new designs and hazards that have not been considered previously. It differs from some of the other techniques described in this chapter in that most of the others require that the hazards be identified before the analysis.

Using a description of the proposed process plant, a HAZOP team (composed of experts on different aspects of the system along with an independent team leader who is an expert on the technique itself) will consider

1. The design intention of the plant
2. The potential deviations from the design intention
3. The causes of these deviations from the design intention
4. The consequences of such deviations

TABLE 14.1
Guidewords for HAZOP.

Guidewords	Meaning
NO, NOT, NONE	The intended result is not achieved, but nothing else happens (such as no forward flow when there should be).
MORE	More of any relevant physical property than there should be (such as higher pressure, higher temperature, higher flow, or higher viscosity).
LESS	Less of a relevant physical property than there should be.
AS WELL AS	An activity occurs in addition to what was intended, or more components are present in the system than there should be (such as extra vapors or solids or impurities, including air, water, acids, corrosive products).
PART OF	Only some of the design intentions are achieved (such as only one of two components in a mixture).
REVERSE	The logical opposite of what was intended occurs (such as backflow instead of forward flow).
OTHER THAN	No part of the intended result is achieved, and something completely different happens (such as the flow of the wrong material).

The guidewords used in this process are shown in Table 14.1. They are applied to any variables of interest such as flow, temperature, pressure, level of composition, and time. Each line in a line drawing of the plant is examined in turn and the guidewords are applied. As each process deviation is generated, the members of the team consider every potential cause (such as a valve closed in error or a filter blocked) and its effect on the system as a whole (such as a pump overheating, a runaway reaction, or a loss of output). Questions are generated from the guidewords. The application of the guideword NONE to flow, for example, which means there should be forward flow, but there is no flow or there is reverse flow, might generate these questions:

- Could there be no flow?
- If so, how could it arise?
- How will the operators know that there is no flow?
- Are the consequences hazardous, or do they prevent efficient operations?
- If so, can we prevent no flow (or protect against the consequences) by changing the design or method of operation?
- If so, does the size of the hazard or problem justify the extra expense?

Figure 14.8 shows a detailed flow chart of the HAZOP process [155] while Table 14.2 shows a typical entry in the table that might result.

The procedure differs for continuous and batch processes. In the HAZOP for a continuous plant, the process is as described above. In addition to normal

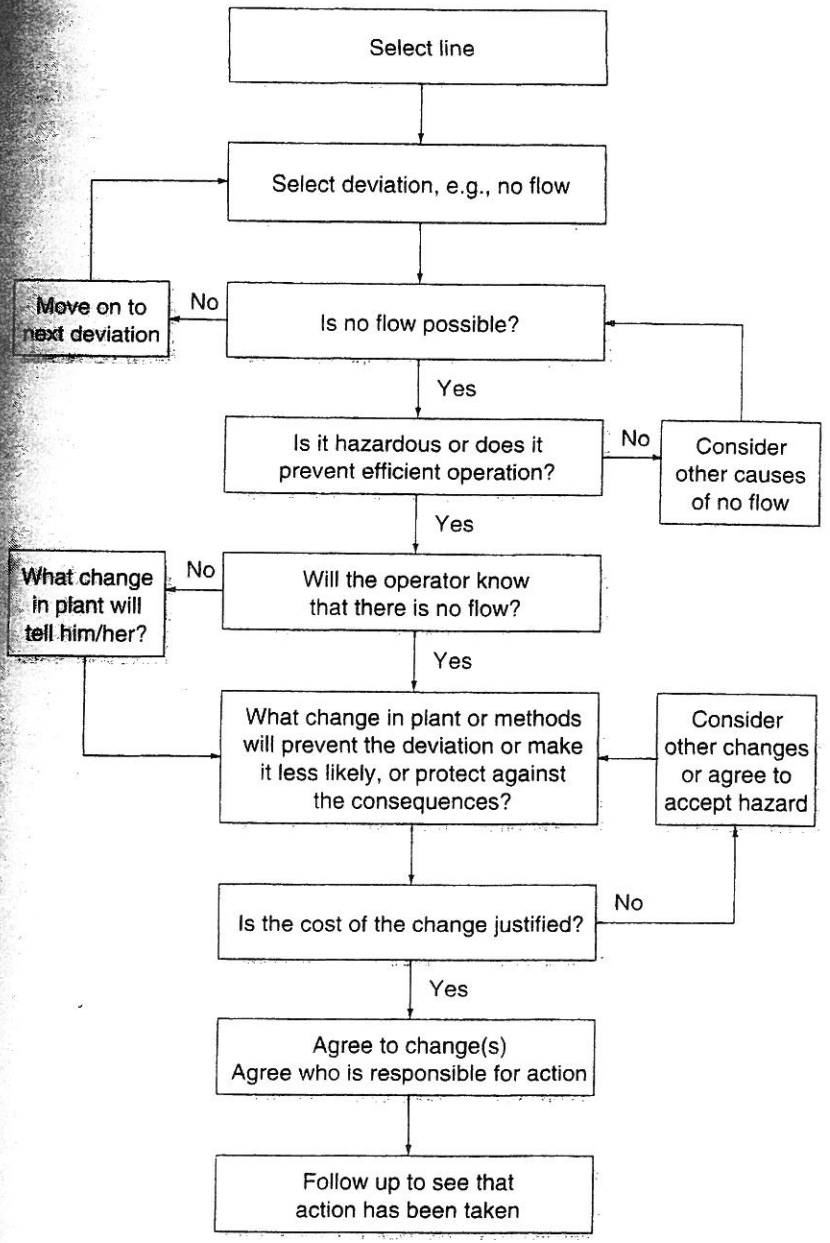


FIGURE 14.8
A flowchart of the HAZOP process. (Source: Trevor A. Kletz, "*Hazop and Hazan*—Notes on the Identification and Assessment of Hazards," Institution of Chemical Engineers, Rugby, U.K., 1983. Reprinted with permission of Trevor Kletz.)

TABLE 14.2
Entry in a HAZOP report.

Guide Word	Deviation	Possible Causes	Possible Consequences
NONE	No flow	1. Pump failure	1. Overheating in heat exchanger
		2. Pump suction filter blocked	2. Loss of feed to reactor
		3. Pump isolation valve closed	

processing, the study should include operability and safety during commissioning of the plant and during regular startup and shutdown.

For a batch plant, not only the flow diagrams but also the operating procedures are examined. The guidewords are applied to the instructions (whether written for operators or executed by a computer) as well as to the pipelines. If computer instructions will be examined, a software engineer should be part of the HAZOP team. Time is important in batch operations: In applying the guidewords to time, such factors as duration, frequency, absolute time, and sequence may be relevant.

Reese has devised an automated variant of HAZOP, called Deviation Analysis, that can be applied to a software requirements specification [290].

Life-Cycle Phase

HAZOP uses process descriptions; flowsheets; control logic diagrams; piping and instrumentation diagrams; a plant layout; draft operating, maintenance, and emergency procedures; safety and training manuals; and data on the chemical, physical, and toxicological properties of all materials, intermediates, and products. By the time this much information is available, it is usually too late to make major changes in the design if hazards are identified. Therefore, hazards are usually controlled by the addition of protection devices rather than removed by design changes [155].

For this reason, many companies conduct preliminary HAZOPs on conceptual flowcharts and preliminary layout diagrams (noting only safety aspects, not operability problems). At this stage, for example, it is possible to replace a flammable piece of equipment with a nonflammable one. At a later stage, when the design is almost complete, it may only be possible to reduce the risk by adding fire insulation, leak detectors, emergency isolation valves, and so on [155]. A full HAZOP usually is conducted later in the design process even if a preliminary HAZOP has been done.

Evaluation

HAZOP does not attempt to provide quantitative results, but instead systematizes a qualitative approach. In most situations, once a hazard is identified, engineering experience or a code of practice is adequate to determine how far to go to remove it. "There is no need, and we do not have the resources, to quantify every hazard on every plant" [161]. In situations where uncertainty remains about the hazard, however, numerical analysis may help to clarify priorities and provide guidance for decision making. In the chemical process industry, the term HAZAN (for HAZard ANalysis) denotes numerical methods.

The strength of the method lies in its simplicity and ease of application and in the early identification of design problems. It does not concentrate only on failures, but has the potential to find more complex types of hazardous events and causes. Reductions of at least an order of magnitude in the number of hazards and problems encountered in operation have been claimed to result from the use of this technique [172].

Although HAZOP is closely connected with the chemical industry, the basic idea could be adapted to other industries (and perhaps has been). HAZOP has the advantage over checklists of being applicable to new designs and design features and of not limiting consideration to previously identified hazards. Complex, potentially dangerous plants with which there is as yet relatively little experience and procedures that occur infrequently (such as commissioning a new plant) are especially good subjects for this type of study [341].

In addition to its open-ended approach to identifying potential problems, a fundamental strength of HAZOP is the encouragement of cross-fertilization of ideas among members of the study team. People from different disciplines working together often find problems that are overlooked by functional groups working in isolation [155]. HAZOP's success, however, depends on the degree of cooperation between individuals, their experience and competence, and the commitment of the team as a whole. Except for the team leader, who is an expert on HAZOP, the members of the team must be experts on the process: The HAZOP procedures allow their knowledge and experience to be applied systematically.

The drawbacks of the technique are the time and effort required—it is labor-intensive—and the limitations imposed by the search pattern. HAZOP relies very heavily on the judgment of the engineers performing the assessment [337]. For example, the extent to which the guideword AS WELL AS is applied will restrict the number of simultaneous faults that can be considered, and evaluation is done by human reasoning alone. Again, all the methods described have these limitations.

Each of the methods has its own search pattern, limiting the factors that will be considered. HAZOP covers hazards caused by process deviations, which is certainly more comprehensive and inclusive than considering failures only, but it still leaves out hazards having more stable determining factors as the only contributors (see Chapter 10) [326]. Examples of causes covered well are failures

of the main operating equipment (such as pumps, compressors, heat exchangers, critical valves, and instrumentation) and human errors in manual operations that involve the main process equipment and its functions (such as opening or closing valves and starting or stopping pumps).

Suokas says that it is unusual for HAZOP to consider deviations or determining factors related to organizational factors such as the information or management systems [327]. On the other hand, an argument can be made that causes related to these factors will be reflected in the process units as a change from the normal state or from acceptable values of the operating parameters, and that tracing back to the causes can reveal management factors. The problem may be more that this type of causal analysis is not encouraged by the technique; the process stops when more proximal factors such as a pump failure are uncovered without necessarily tracing the failure back to a maintenance error and perhaps back from that to a management problem.

14.8 Interface Analyses

Description

Various analysis methods are used to evaluate connections and relationships between components, including incompatibilities and the possibilities for common-cause or common-mode failure. In general, the relationships examined can be categorized as physical, functional, or flow [108]. These analysis methods generally use structured walkthroughs to examine the interface between components and to determine whether a connection provides a path for failure propagation. The types of problems and effects that are examined include

- No output from the unit or interconnection failures that cause the receiving unit not to receive the output of the upstream unit.
- Degraded output or partial failures of the unit or interconnection.
- Erratic output (intermittent or unstable operation).
- Excessive output.
- Unprogrammed output (inadvertent operation or erroneous output).
- Undesired side effects (programmed outputs are within specified limits, but additional damaging outputs are produced), such as a unit generating heat that can shorten the lives of nearby units.

Any such analysis should include connections between components that go through the software.

Noble has defined a specialized version of interface analysis that considers the potential for common-mode failures to affect redundant hardware components. His hardware and software common-mode failure analysis examines each connection between redundant components (including connections through software) to determine whether the connection provides a path for failure propagation [241].

Evaluation

Interface analyses are similar to HAZOP, but generalized somewhat, so they have the same benefits and limitations. Effectiveness depends upon the procedures used and the thoroughness with which the analysis is applied.

14.9 Failure Modes and Effects Analysis

Description

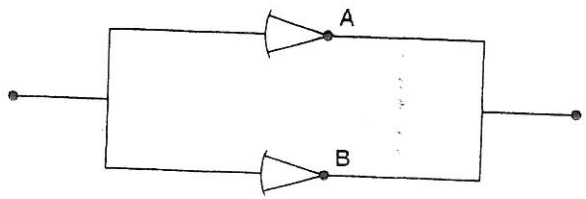
Failure Modes and Effects Analysis (FMEA) was developed by reliability engineers to permit them to predict equipment reliability. As such, it is a form of reliability analysis that emphasizes successful functioning rather than hazards and risk. The goal is to establish the overall probability that the product will operate without a failure for a specific length of time or, alternatively, that the product will operate a certain length of time between failures.

Like event trees, FMEAs use forward search based on an underlying chain-of-events model, where the initiating events are failures of individual components. The first step in an FMEA is to identify and list all components and their failure modes, considering all possible operating modes. For each failure mode, the effects on all other system components are determined along with the effect on the overall system. Then the probabilities and seriousness of the results of each failure mode are calculated.

Component failure rates are predicted from generic rates that have been developed from experience and are often published. Information centers collect and collate such information, and manufacturers usually have this data for their own products. Care must be taken that the environment in which the component will be working is identical to the one for which the statistics were collected. Probabilities are based on averages collected over large samples, but individual components may differ greatly from the average, perhaps because of substandard manufacturing or extreme environments. Confidence levels and error bounds are often omitted from FMEAs, but should be included.

The results are documented in a table with column headings such as component, failure probability, failure mode, percent failures by mode, and effect (which may be broken down into critical and noncritical or any other categories desired).

Figure 14.9 shows a simple FMEA for two amplifiers in parallel [344]. This example assumes that an amplifier failing short or in some other mode causes the system to fail while an amplifier failing open does not. The probabilities in the



Critical	Failure probability	Failure mode	% failures by mode	Effects	
				Critical	Noncritical
A	1×10^{-3}	Open	90		X
		Short	5	5×10^{-5}	
		Other	5	5×10^{-5}	
B	1×10^{-3}	Open	90		X
		Short	5	5×10^{-5}	
		Other	5	5×10^{-5}	

FIGURE 14.9
FMEA for a system of two amplifiers in parallel. (Source: W. E. Vesely, F. F. Goldberg, N. H. Roberts, and D. F. Haasl, *Fault Tree Handbook*, NUREG-0492, U.S. Nuclear Regulatory Commission, Washington, D.C., 1981, page II-3)

column labeled *critical effects* (that is, they cause system failure) are added to get a failure probability for the entire system.

Life-Cycle Phase

FMEAs are appropriate when a design has progressed to the point where hardware items may be easily identified on engineering drawings and functional diagrams. The analyst needs a detailed design that includes schematics, functional diagrams, and information about the interrelationships between component assemblies.

Evaluation

FMEA is effective for analyzing single units or single failures to enhance individual item integrity. It can be used to identify redundancy and fail-safe design

requirements, single-point failure modes, and inspection points and spare part requirements. It is also useful in determining how often the system must be serviced and how components and designs must be improved in order to extend the operational life of a product.

The strength of the technique is its completeness, but that means it is also very time consuming and can become tedious and costly if applied to all parts of a complex design.

All the significant failure modes must be known in advance, so FMEA is most appropriate for standard parts with few and well-known failure modes. The technique itself does not provide any systematic approach for identifying failure modes or for determining their effects and no real means for discriminating between alternate courses of improvement or mitigation. In fact, for systems that exhibit any degree of complexity, identifying all possible component failure modes—both singly and in combination—becomes simply impossible [344].

FMEA does not normally consider effects of multiple failures; each failure is treated as an independent occurrence with no relation to other failures in the system except for the subsequent effects it might produce. By limiting the analysis to single units and not considering multiple- or common-cause failures, the technique becomes simple to apply and the examination is very orderly, but the results may be of limited use if time sequences and the interrelationships among the elements of a complex system are not considered. Studies of product failures have shown that a much greater number are the result of connector problems than of failures in the components themselves [108].

Hammer points out that, as usually applied, FMEAs pay little attention to human errors in operating procedures, hazardous characteristics of the equipment, or adverse environments [106]. Although environmental conditions are considered in identifying the stresses that could cause hardware to fail, the probabilities of occurrence of such environmental stresses are rarely used. Instead, a usage factor is incorporated for the type of system application, such as shipboard, aircraft, or missile use, and another factor is applied for reduction of theoretical reliability that could result from substandard manufacture or assembly. This latter factor is extremely rough, even over a large sample. "Oddly enough," Hammer says, "in spite of all those factors affecting a system but whose probability of occurrence can only be estimated imprecisely, reliability engineers carry out their calculations to six or seven significant figures."

Because they establish the end effects of failures, FMEAs are sometimes used in safety analyses. If the limitations are understood, there is no problem with this. Not all failures result in accidents, however, so analyzing all parts, the ways each part can fail, and the resultant effects is generally a time-consuming and inefficient way to obtain safety-related information. In addition, the technique provides only a small part of the information needed, since the probability of damage determined by an FMEA is related to individual failures only; it rarely involves investigating damage or injury that could arise if multiple components fail or if the components operate successfully.

14.10 Failure Modes, Effects, and Criticality Analysis

Description

Failure Modes, Effects, and Criticality Analysis (FMECA) is basically just an FMEA with a more detailed analysis of the criticality of the failure. Two additional steps (and usually columns) are added to the FMEA: (1) the means of control already present or proposed are determined, and (2) the findings are modified with respect to these control procedures (such as modifying the chance of failure or adding an indication of whether or not further control is necessary). An example is shown in Figure 14.10.

Criticality rankings are generally expressed as probabilities or frequencies, such as the number of failures of a specific type expected during each 1 million operations performed in a critical mode. Rankings may also be ordered in categories from 1 to 10, or assigned letters starting from the beginning of the alphabet [108], to show the principal items that generate problems.

Along with the ranking, a description is provided of the preventive and corrective measures that should be taken and the safeguards to be incorporated.

Sometimes a Critical Items List (CIL) is generated from the results of the FMEA or FMECA. This list might include item, list of possible failure modes, failure probability (for each mode), effect on the mission (such as abort, degradations of performance, or damage) and criticality ranking within the subsystem (perhaps using a numerical scale).

Evaluation

Since this technique is simply an FMEA with two columns added, the same evaluation applies, with the exception that the FMECA does include a description of the means of controlling the failure. Even more effort is now required, though, and it still does not consider one aspect of criticality—possible damage. Hammer [108] suggests various ways that damage could be incorporated.

14.11 Fault Hazard Analysis

Description

Fault Hazard Analysis (FHA) was developed about the same time as FTA and was also used on the Minuteman missile system [344]. It is basically a FMEA or FMECA with both a broader and more limited scope. The scope is broadened by considering human error, procedural deficiencies, environmental conditions, and other events that might result in a hazard caused by normal operations at an undesired time [69]. At the same time, its scope is more restricted than that of a FMEA or FMECA, since supposedly only failures that could result in accidents

Failure Modes and Effects Criticality Analysis

Subsystem _____ Prepared by _____ Date _____

Item	Failure Modes	Cause of Failure	Possible Effects	Prob.	Level	Possible Action to Reduce Failure Rate or Effects
Motor Case	Rupture	a. Poor workmanship b. Defective materials c. Damage during transportation d. Damage during handling e. Overpressurization	Destruction of missile	0.0006	Critical	Close control of manufacturing processes to ensure that workmanship meets prescribed standards. Rigid quality control of basic materials to eliminate defectives. Inspection and pressure testing of completed cases. Provision of suitable packaging to protect motor during transportation.

FIGURE 14.10
A sample FMECA.

are considered, although it is difficult to understand how a forward analysis of this type can be done without all failures being considered first.

Two new pieces of information are added about upstream and downstream effects: (1) upstream components that could command or initiate the fault in question and (2) factors that could lead to secondary failures. The effects on the system are briefly stated in terms of associated damage or malfunction. The column headings may include component, failure probability, failure modes, percent failure by mode, effect of failure (traced to some relevant interface), upstream components that could command or initiate the failure or fault, and factors that could cause secondary failures.

Evaluation

Like FMEAs and FMECAs, FHA primarily provides guidance on what information to obtain, but it provides no help in actually getting that information. And again, in use FHA tends to concentrate primarily on single events or failures.

The technique was developed as a special tool for use on projects involving many organizations, one of which acts as an integrator. Hammer says that FHA is useful in considering faults that cross organizational interfaces. Others consider the technique to have little use.

14.12 State Machine Hazard Analysis

Description

A state machine is a model of the states of a system and the transitions between them. Figure 14.11 shows a simple state machine model of a level control. The model has three states (represented by circles): water level low, water level high, and water level at the set point. The arrows represent transitions between states. Each arrow has the condition for changing state and an output action attached to it. When a condition on a transition from a state becomes true and the machine is in that state, the machine changes to the new state and takes the output action. In the example, depending on the sensor reading of the water level and the current state of the machine, the machine will activate the pump, turn off the pump, open the drain, or close the drain.

State machine models are used often in computer science. One of the problems with using them for complex systems is the large number of states that these systems have and thus must be specified. One way to avoid this problem is to use models that abstract away from all the states to a smaller number of higher-level states, from which the entire state machine can be generated. The complete "state space" may never be generated (and it may be infeasible to do so), but many properties of the state space can be inferred from the higher-level model.

With respect to safety, if a model of the system to be built were created and its entire state space generated, it would be possible to determine if the state

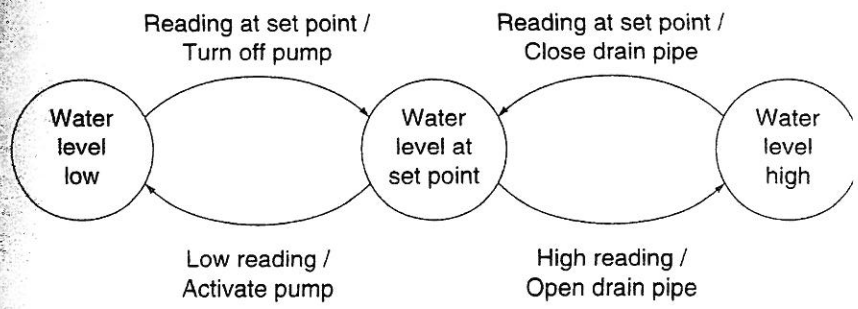


FIGURE 14.11
A state machine model of a water level control.

space contained any hazardous states. Basically, this approach involves a fo ward search that starts from the initial state of the system, generates all possibl paths from that state, and determines whether any of them are hazardous. Unfo tunately, for most realistic systems, the computational effort involved makes thi approach impractical, even if computers are used.

Backward and top-down search in the general sense is also impractical. would entail starting with the hazardous states and working backward from eac to see if the initial state is reached. If so, then the hazardous state is reachable an the model is unsafe. If not, then the hazardous state is not reachable. The numbe of backward paths is still enormous for real systems, even if only those ending i hazardous states are considered.

A practical solution is to start from the hazardous state and only work fa enough back along the paths to determine how to change the model to mak the hazardous state unreachable [186]. Only a small number of the states wi need to be generated in most cases. The drawback, although not a serious on is that the hazardous states eliminated from the design might not actually hav been reachable, so more hazards may be eliminated than were actually presen This algorithm was first demonstrated using a Petri-net model,[1] but the procedur can be adapted for any state machine model. Any parts of the system that ca be modeled using state machines can be included in the hazard analysis. If fault and failures are included in the model, their effect on the system behavior can b determined.

[1] A Petri net is a mathematical representation of a discrete event system that is especially appropriate for representing systems with interacting, concurrent components. Petri nets model discrete state systems in terms of conditions and events and the relationship between them. Algorithms exist for generating the reachable states from a Petri-net model although this procedure may not be practical for all systems. See Peterson [262] for a complete description of Petri nets.

State Machine Hazard Analysis (SMHA) was first developed to identify software-related hazards [186]. Software and other component behavior is modeled at a high level of abstraction, and faults and failures are modeled at the interfaces between the software and the hardware; thus, the procedure can be performed early in the system and software development process.

SMHA can be used to analyze a design for safety and fault tolerance, to determine software safety requirements (including timing requirements if the model includes timing) directly from the system design, to identify safety-critical software functions, and to help in the design of failure detection and recovery procedures and fail-safe requirements. Since the model used is formal (that is, it has a mathematical definition), the analysis procedures can be implemented on a computer.

Life-Cycle Phase

SMHA works on a model, not the design itself. Therefore, it can theoretically be used at any stage of the life cycle, including early in the conceptual stage, to evaluate alternative designs and design features. The procedure is most effective if performed before the detailed design of the system components begins.

Evaluation

SMHA can be carried out before detailed design of the system is finished, although the partitioning of functions to components must be at least tentatively complete. Since the analysis is performed on a formal, written model, it can be automated and does not depend on the analyst's mental model of how the system works. The model is explicitly specified and can be checked for correctness by expert review and sometimes for various desirable properties by additional automated procedures. Often, the state machine model itself can be executed using test data and simulators.

SMHA's most important limitation is that a model must be built, which may be difficult and time consuming. State machine models have been built for parts of systems and for relatively small systems, but are often impractical for systems that are large or complex. Petri nets, on which the algorithms were first defined, are not a practical modeling language for most real systems. Recent advances in state machine modeling languages have overcome this problem somewhat by defining new types of higher-level abstractions [111]. These abstractions have been incorporated into several languages, one of which, Requirements State Machine Language (RSML), was adopted by the FAA to model the system requirements for TCAS II, an airborne collision avoidance system required on most aircraft in the United States [360].

The SMHA analysis algorithms have been adapted for the RSML language and are being applied experimentally to real systems. Work is also proceeding on automatically generating fault trees and additional standard hazard analysis models from the RSML specification. Other new state machine models could be,

but have not been, used for safety analysis, nor have safety analysis procedure been defined for them.

Some of the effort in building the model is justified by the fact that it can be used as the system requirements specification. To be used for this purpose the specification must be readable by people without advanced mathematical education. The mathematical model analyzed by the SMHA algorithms is actually generated from the high-level RSML specification language, which is readable by application experts with very little training. RSML was developed while specifying the system requirements for TCAS II, which had to be easily readable and reviewable by engineers, pilots, airline representatives, and others in its function as the FAA system specification. The RSML specification can also be simulated (both general and application-specific simulators have been built) so the model can be executed, and test data (for the later software implementation of the specification) can be generated from it. The practicality of the SMHA analysis procedures for RSML has yet to be verified, however, and though the analysis procedures have been experimentally applied to the TCAS II specification, they have not yet been used on other projects.

A second limitation of SMHA is that the analysis is performed on a model, not on the system itself—it will apply to the as-built system only if the system matches the model. This limitation holds, of course, for any analysis that is performed early in the life cycle, but appropriate design and verification procedures must be used to ensure that the implemented system matches the model on which the analysis was originally performed.

Other types of mathematical models, such as logic or algebraic models of software or systems, also could be used for hazard analysis by using mathematical proof methods to show that the models satisfy the safety requirements [137, 283]. Many logic and algebraic models and modeling languages have been proposed for software. Unfortunately, most have been tried only on very small examples, and it is not at all clear that they will scale up to realistic systems. In addition, writing down the model may not be as much of a problem as the effort involved in mathematically proving the safety properties of the system and the inability of reviewers to understand those proofs.

The most important limitation of these algebraic and logic languages is that they are usually very hard to learn and use (including performing proofs on them) without an advanced degree in mathematics. This factor by itself is not necessarily a problem, as people with such training exist or the training can be provided, but the resulting models and proofs cannot readily be understood or checked by engineers and application experts who do not have this training. One of the most important uses of any hazard analysis is as an aid for designers and as a representation of the problem and what is being done about it so that open discussion can be stimulated and supported. If the analysis cannot be audited and understood by application experts, confidence in the results is undermined.

In addition, the models and languages used must match the way that engineers think about the systems they are building, or the translation between the engineer's or expert's mental model and the written formal model will be error

prone. The advantage of state machine models is that they seem to match the internal models many people use in trying to understand complex systems.

14.13 Task and Human Error Analysis Techniques

14.13.1 Qualitative Techniques

Much more emphasis in hazard analysis has been on equipment failures than on human errors. Some analysis methods for human error have been suggested, however, including Procedure (Task) Analysis, Operator Task Analysis, Work Safety Analysis, and Action Error Analysis.

A procedure is an ordered set of instructions or actions to accomplish a task. *Procedure* or *Task Analysis* [106] reviews procedures to verify that they are effective and safe within the context of the mission tasks, the equipment that must be operated, and the environment in which the personnel must work. Such analyses involve determination of the required tasks, exposures to hazards, criticality of each task and procedural step, equipment characteristics, and mental and physical demands. As with FMEAs, the results of the analysis are entered on a form with columns labeled Task, Danger, Effects, Causes, Corrective Measures, and so on. Possible results include recommendations for corrective or preventive measures to minimize the possibilities that an error will result in a hazard, changes or improvements in hardware or procedures, warning and caution notes, special training, and special equipment (including protective clothing).

Operator Task Analysis [172] appears to be another name for Procedure Analysis. The operator's task is broken down into separate operations, and the analysis looks for difficulties in executing either the individual operations or the overall plan. Neither of these first two analyses (Procedure Analysis and Operator Task Analysis) seems to have a specific procedure associated with it, and they may simply be generic terms for the goals involved.

Action Error Analysis (AEA) [323, 326] uses a forward search strategy to identify potential deviations in human performance. The analysis consists of a systematic description of the operation, task, and maintenance procedures along with an investigation of the potential for performance deviations (such as forgetting a step, wrong ordering of steps, and taking too long for a step). Internal phases of data processing associated with an operator's tasks are usually excluded; instead, only the external outcomes of the error modes in different steps are studied. Some information about physical malfunctions may result from the analysis, since it includes the effects of human malfunctions on the physical equipment. This method is very similar to FMEA, but is applied to the steps in human procedures rather than to hardware components or parts. The results are entered in a table, this time with columns labeled Work Step, Action Error, Primary Consequences, Secondary Consequences, Detection, and Measures.

Work Safety Analysis (WSA) [323, 342] was developed by Suokas and Rouhiainen in Finland in the early 1980s. It is similar to HAZOP, but the search process is applied to work steps. The goal is to identify hazards and their causes. The search starts, as in the other methods, by breaking a task down into a sequence of steps. Each of the steps is examined with respect to a list of general hazards and examples of their causes (deviations and determining factors). All types of system functions and states, including normal states, are considered. The analyst examines the consequences of (1) forgetting a work step, (2) performing a step too early or too late or too long, and (3) unavailability of the usual equipment. Because of the nature of the search pattern, certain types of hazards will not be identified, such as those related to management procedures or those related indirectly to the operator's task but not to the task being analyzed (for example, contact with chemicals or an explosion in the proximity of the operator) [326].

14.13.2 Quantitative Techniques

All the human error analysis methods described so far focus on the operator's task. The goal is to obtain the information necessary to design a human–machine interface that reduces human behavior leading to accidents and improves the operators' ability to intervene successfully to prevent accidents. Human error is not considered inevitable, but a result of human–task mismatches and poor interface or operating procedures design. When the focus is design, qualitative or semi-quantitative results are usually adequate to achieve the goals.

Probabilistic assessment of human error, on the other hand, necessarily accepts the inevitability of human error. Despite its limited usefulness in improving the human–machine interface, the application of reliability engineering, which focuses on numerical assessment, to process control systems (especially nuclear power plants) has led to a demand for assessing the reliability of the process operator in order to assess risk for the system as a whole. The assignment of probabilities to human error is especially important in system risk assessment because of the large proportion of accidents that are attributed to human error.

Simply having a need is not enough to guarantee that the need can be satisfied. Probabilistic assessment of human error is not very advanced. Some of the problems in collecting and classifying human error data were discussed in Chapter 13. This rest of this section describes the current state of the art; readers can determine for themselves how much confidence they want to place on the resulting numbers.

Most of the numerical data and assessment are based on task analysis and task models of errors rather than on cognitive models. Following Lees' classification (see Chapter 10), tasks are divided into simple, vigilance, and complex.

Simple and Vigilance Tasks

Simple tasks are relatively simple sequences of operations involving little decision making. Some of these tasks or suboperations may involve the detection of signals (vigilance).

The most common way to assign probabilities to these tasks is to break

a task down into its constituent parts, assign a reliability to the execution of each part, and then estimate the reliability of the entire task by combining the reliability estimates of the parts using a structural model of their interaction. The most common models involve either series relationships (and thus use product laws) or tree relationships (and use Boolean evaluation methods). The accuracy of the method depends upon the accuracy of the individual part reliabilities and the appropriateness of the structural model.

The sophistication of the quantitative reliability estimates varies greatly [172]. The simplest approaches often use an average task error rate of 0.01. This number is based on the assumption that the average error rate of the constituent task components is 0.001 and that there are, on average, 10 components per task.

A second approach to assigning human error rates uses human experts to rank tasks in order of their error likeliness and then uses ranking techniques to obtain error rates. Sophisticated statistical methods, such as paired comparisons, can be used to produce a ranking [130].

The techniques described so far rely on human judgment to assign error rates to tasks, or they make very simple assumptions. Other approaches collect and use empirical and experimental data evaluated with respect to performance-shaping factors. *Data Store* was developed by the American Institute for Research in 1962 to predict operator performance in the use of controls and displays [108]. The data indicates the probability of successful performance of a task, the time required to operate particular instruments, and the features that degrade performance. To analyze a task using Data Store, the task components are identified and assigned probabilities using tables for standardized tasks. The reliabilities are then multiplied to determine a task reliability.

Data Store and similar techniques assume that the discrete task components are independent. THERP (Technique for Human Error Rate Prediction), developed by Swain at Sandia National Laboratories, relaxes this assumption. Bell and Swain describe a methodology for Human Reliability Analysis (HRA) that encompasses both task analysis and THERP [22].

Most of the errors identified and analyzed in HRA involve not following written, oral, or standard procedures. Only occasionally are actions that are outside the scope of the specified operations (such as extraneous acts) considered.

The first part of HRA (and of most similar methods) involves task analysis, where a task is defined by Bell and Swain as a quantity of activity or performance that the operator views as a unit, either because of its performance characteristics or because the activity is required as a whole to accomplish some part of a system goal. The correct procedure for accomplishing an operation is identified and then broken down into individual units of physical or mental performance. For example, the tasks involved in pressurizing a tank to a prescribed level from a high-pressure source [106] include

1. Opening the shutoff valve to the tank
2. Opening the high-pressure regulator from the source

E 14.3
el human error data.

ability	Activity
10^{-2}	General human error of omission where there is no display in the control room of the status of the item omitted, such as failure to return a manually operated test valve to the proper position after maintenance.
3×10^{-3}	Error of omission where the items being omitted are embedded in a procedure rather than at the end.
3×10^{-2}	General human error of commission, such as misreading a label and therefore selecting the wrong switch.
3×10^{-2}	Simple arithmetic error with self-checking, but without repeating the calculation by redoing it on another piece of paper.
10^{-1}	Monitor or inspector failure to recognize an initial error by operator.
10^{-1}	Personnel on different workshift fail to check the condition of hardware unless required by a checklist or written directive.

3. Observing the pressure gauge downstream from the regulator until the prescribed level is reached in the tank
4. Shutting off the high-pressure regulator
5. Shutting the valve to the tank

Next, specific potential errors (human actions or their absence) are identified for each unit of behavior in the task analysis. Acts of commission and omission are considered errors if they have the potential for reducing the probability of some desired system event or condition. In the above example, the operator could forget to open the high-pressure regulator from the source (step 2), open the wrong valve (step 1), or execute the actions out of proper sequence. The actions actually considered are limited. For example, if the error being examined is the manipulation of a wrong switch, perhaps because of the control panel layout, the analysis does not usually try to predict which other switch will be chosen, nor does it deal with the system effects of the operator selecting a specific incorrect switch.

The next step in HRA is to determine the likelihood of specific event sequences using event trees. Each error defined in the task analysis is entered on the tree as a binary event. If order matters, then the events need to be ordered chronologically. Care must be taken to consider all alternatives, including "no action taken." Other logical models, such as fault trees, can also be used.

Probabilities are assigned to each of the events in the tree, using handbooks or tables of human error probabilities. If an exact match of errors is not possible, similar tasks are used and extrapolations are made. Table 14.3 is a small example of this type of table [172].

The data in the THERP handbook is based on a set of assumptions that limit the applicability of the data [22]:

- The operator's stress level is optimal.
- No protective clothing is worn.
- The level of administrative control is average for the industry.
- The personnel are qualified and experienced.
- The environment in the control room is not adverse.
- All personnel act in a manner they believe to be in the best interests of the plant (malevolent action is not considered).

Because these assumptions may not hold and because of natural variability in human performance, environmental factors, and task aspects, the THERP handbook gives a best estimate along with uncertainty bounds. The uncertainty bounds represent the middle 90 percent range of behavior expected under all possible scenarios for a particular action; they are based on subjective judgment rather than empirical data. The analyst is expected to modify the probabilities used in HRA to reflect the actual situation. Examples of performance shaping factors that can affect error rates are

- Level of presumed psychological stress
- Quality of human engineering of controls and displays
- Quality of training and practice
- Presence and quality of written instructions and methods of use
- Coupling of human actions
- Personnel redundancy (such as the use of inspectors)

Bell and Swain [22] suggest that if, for example, the labeling scheme at a particular plant is very poor compared to labeling at other plants, the probabilities should be increased toward the upper uncertainty bound;[2] if the tagging is particularly good, the probabilities for certain errors might be decreased. These performance shaping factors either affect the whole task or affect certain types of errors regardless of the types of tasks in which they occur. Other factors may have an overriding influence on the probability of occurrence of all types of errors under all conditions.

Dependencies or coupling may exist between pairs of tasks or between the performance of two or more operators. The dependencies in the specific situation need to be assessed and estimates made of the conditional probabilities of success and failure.

Once all these steps have been accomplished, the end point of each path through the event tree can be labeled a success or a failure, and the probability of each path can be computed by multiplying the probabilities associated with each path segment. Then the success and failure probabilities of all the paths are combined to determine the total system success and failure probabilities. The results of HRA are often used as input to fault trees and other system hazard

[2] The system safety engineer might suggest instead that the labeling at the plant be improved.

TABLE 14.4
Typical error rates used for emergency situations.

Probability	Activity
0.2 – 0.3	The general error rate given very high stress levels where dangerous activities are occurring rapidly.
1.0	Operator fails to act correctly in first 60 seconds after the onset of an extremely high stress condition.
9×10^{-1}	Operator fails to act correctly in the first 5 minutes after the onset of an extremely high stress condition.
10^{-1}	Operator fails to act correctly in the first 30 minutes of an extreme stress condition.
10^{-2}	Operator fails to act correctly in the first several hours of a high stress condition.

analyses, although care must be taken that the limitations and assumptions are not violated.

Humans make errors, but they also often detect their errors and correct them before they have a negative effect on the system state. If it is possible to recover from an error in this way, the actual error rate for the task may be reduced by orders of magnitude from the computed rate [172]. The probability of recovery depends greatly on the cues available to the operator from the displays and controls and from the plant in general. Bell and Swain suggest that the effects of recovery factors in a sequence of actions not be considered until after the total system success and failure probabilities are determined. These may be sufficiently low, without considering the effects of recovery, so that the sequence does not represent a dominant failure mode. Sensitivity analyses (manipulating a particular parameter to determine how changes to its value affect the final value) can also be performed to identify errors that have a very large or very small effect on system reliability.

Most of these probabilities do not apply to tasks under emergency conditions, where stress is likely to be high. Analyses usually assume that the probability of ineffective behavior during emergencies is much greater than during normal processing. In general, error probability goes down with greater response time. For short response times, very little credit is normally given for operator action in an emergency. Table 14.4 shows some typical error rates used for emergency situations [172].

One other factor needs to be considered when computing or using these numbers, and that is sabotage or deliberate damaging actions by the operator, including suicide. Most of the available data on human behavior assumes that the operator is not acting malevolently; instead it assumes that any intentional deviation from standard operating procedures is made because employees believe their method of operation to be safer, more economical, or more efficient, or because they believe the procedure is unnecessary [22]. Ablitt, in a UK Atomic

Energy Authority publication discusses the possibility of suicide by destruction of a nuclear power plant:

> The probability per annum that a responsible officer will deliberately attempt to drop a fuel element into the reactor is taken as 10^{-3} since in about 1000 reactor operator years, there have been two known cases of suicide by reactor operators and at least one case in which suicide by reactor explosion was a suspected possibility. The typical suicide rate for the public in general is about 10^{-4} per year although it does vary somewhat between countries (quoted in [172, p.411]).

Other human reliability estimation techniques have been proposed, although THERP is probably the most widely used. A weakness of all these techniques, as noted, is that they do not apply to emergency situations (very little data on human errors in emergencies is available). If one accepts Rasmussen's Skill–Rule–Knowledge model, the error mechanisms embedded in a familiar, frequent task and in an infrequent task will differ because the person's internal control of the task will be different [270]. Therefore, error rates obtained from general error reports will not apply for infrequent responses.

Another weakness is that the techniques cannot cope with human decisions and tasks that involve technical judgment. Factors other than immediate task and environmental factors are also ignored.

Embrey [77] has suggested an approach to investigating human mistakes linked to organizational weaknesses. His *Goal Method* relates the goals of an operator responsible for specific equipment to the goals of the plant as a whole. Hope and colleagues [126] say that this approach is helpful in training operating teams, particularly for emergency situations.

Many of these human reliability assessment techniques were proposed and the data collected before plants became highly automated, especially by computers. We are automating exactly those tasks that can be measured and leaving operators with the tasks that cannot. Therefore, measurement of this type is bound to be of diminishing importance.

Complex Control Tasks

The measurement approaches described in the previous section consider human performance as a concatenation of standard actions and routines for which error characteristics can be specified and frequencies determined by observing similar activities in other settings. In such analyses, the task is modeled rather than the person. Rasmussen and others argue that such an approach may succeed when the rate of technological change is slow, but is inadequate under the current conditions of rapid technological change [278].

Computers and other modern technology are removing repetitive tasks from humans, leaving them with supervisory, diagnostic, and backup roles. Tasks can no longer be broken down into simple actions; humans are more often engaged in decision making and complex problem solving for which several different paths may lead to the same result. Only the goal serves as a reference point when judging the quality of performance—task sequence is flexible and very situation and person specific. Analysis, therefore, needs to be performed in terms of the cognitive information processing activities related to diagnosis, goal evaluation priority setting, and planning—that is, in the knowledge-based domain.

From this viewpoint, performance on a task can no longer be assumed to be at a relatively stable level of training. Learning and adaptation during performance will have a significant impact on human behavior. If the models of behavior used do not merely consider external characteristics of the task but have a significant cognitive component, then measurement (and, of course, design) needs to be related to internal psychological mechanisms in terms of capabilities and limitations [274]. If, as Rasmussen recommends, the concept of human error is replaced by human–task mismatch, then task actions cannot be separated from their context. Rasmussen suggests that a FMEA can serve as a basis for analysis of a human–task mismatch. Numbers for these models do not exist and deriving them will be difficult, however, as the cognitive activities involved in complex and emergency situations cannot easily be identified in incident reports. Top-down analysis can also be used (and seems more promising) to relate critical operator errors to cognitive human error models.

14.14 Evaluations of Hazard Analysis Techniques

Given the widespread use of hazard analysis techniques, the small amount of careful evaluation is surprising. The techniques are often criticized as incomplete and inaccurate, but this criticism is based on logical argument rather than on scientific evaluation. Only a few critical evaluations of hazard analysis methods have been performed, and most simply evaluate the structure of the methods. Taylor, Suokas, and Rouhiainen, however, have actually performed empirical evaluations.

Taylor applied HAZOP and AEA to two plants and compared the results with problems found during commissioning and a short operating period. HAZOP found 22 percent and 80 percent of the hazards, while the corresponding results for AEA were 60 percent and 20 percent for the two analyses evaluated [322].

Suokas compared HAZOP to AEA, WSA, and accident investigations for two gas storage and loading–unloading systems. HAZOP identified 77 contributors to a gas release. AEA and WSA found 23 additional factors not found by HAZOP. When the results were quantified with fault trees, the contributors identified only by AEA increased the total frequency of gas release by 28 percent in one system and by 38 percent in the other [327].

Suokas and Pyy evaluated four methods—HAZOP, FMEA, AEA, and MORT—by collecting incident and accident information in seven process plants and one accident database. They defined the search patterns and types of factors

covered by the methods, and three groups evaluated which of the causal factors the methods could have identified. HAZOP was the best, identifying 36 percent of the contributors. However, only 55 percent of the contributors were expected to be covered by the four methods [323, 322]. This result is particularly poor given that the analysis involved only determining which factors *could* potentially be identified by the methods—the number actually identified in any application would be expected to be lower.

Many evaluations of the predictive accuracy of reliability estimates have been done for individual instruments and components; these studies vary widely in their results. In a reliability benchmark exercise, 10 teams from 17 organizations and from 9 European countries performed parallel reliability analyses of a nuclear power plant primary cooling system. The purpose was to determine the effect of differences in modeling and data. The ratio between the highest and lowest frequencies calculated for the top event of the different fault trees was 30. When a unified fault tree was quantified by different teams using what each considered to be the best data, the corresponding ratio was reduced to 9.

14.15 Conclusions

Many different hazard analysis techniques have been proposed and are used, but all have serious limitations and only a few are useful for software. But whether these techniques or more ad hoc techniques are used, we need to identify the software behaviors that can contribute to system hazards. Information about these hazardous behaviors is the input to the software requirements, design, and verification activities described in the rest of this book.

Chapter

15

Software Hazard and Requirements Analysis

Computers do not produce new sorts of errors. They merely provide new and easier opportunities for making the old errors.

—Trevor Kletz
Wise After the Event

The vast majority of accidents in which software was involved can be traced to requirements flaws and, more specifically, to incompleteness in the specified and implemented software behavior—that is, incomplete or wrong assumptions about the operation of the controlled system or required operation of the computer and unhandled controlled-system states and environmental conditions. Although coding errors often get the most attention, they have more of an effect on reliability and other qualities than on safety [80, 200].

This chapter describes completeness and safety criteria for software requirements specifications. The criteria were developed both from experience in building such systems and from theoretical considerations [135, 136] and, in essence, are the equivalent of a requirements safety checklist for software. They can be used to develop informal or formal inspection procedures or tools for automated analysis of specifications. The criteria are general and apply to all systems, unlike the application-specific safety requirements identified in a system hazard analysis. Both application-specific hazards and general criteria need to be checked—in fact, one of the general criteria requires checking the application-specific hazards.