SAFEWARE

System Safety and Computers

JUNBEOM YOO

Dependable Software Laboratory
KONKUK University

http://dslab.konkuk.ac.kr

Ver. 1.0 (2011.03.07)

Contents

Part One: The Nature of Risk

1. Risk in Modern Society
2. Computers and Risk
3. A Hierarchical View of Accidents
4. Root Causes of Accidents
5. Human Error and Risk
6. The Role of Humans in Automated Systems

Part Two: Introduction to System Safety

7. Foundations of System Safety
8. Fundamentals of System Safety

Contents

Part Three: Definitions and Models

9. Terminology
10. Accident and Human Error Models

Part Four: Elements of a Safeware Program

11. Managing Safety
12. The System and Software Safety Process
13. Hazard Analysis
14. Hazard Analysis Models and Techniques
15. Software Hazard and Requirements Analysis
16. Designing for Safety
17. Design of the Human-Machine Interface
18. Verification of Safety

Part One

THE NATURE OF RISK

A life without adventure is likely to be unsatisfying, but a life in which adventure is allowed to take whatever form it will, is likely to be short.

- Bertrand Russell

Chapter 1

RISK IN MODERN SOCIETY

Introduction

- Is new technology making our world riskier?

- Risk is not a new problem.
  - Humans have always had to face risk from their environment.
  - The risks have changed as society and the natural environment have changed.

- The distinction between natural and technological hazards is often useful, but is not completely accurate.
  - In fact, all hazards are affected by complex interactions among technological, ecological, sociopolitical, and cultural systems [30,174,339].
  - Attempts to control risk by treating it simply as a technical problem or only as a social issue are doomed to fail or to be less effective than possible.

- To discover effective solutions to these problems,
  - We must understand the factors underlying the risks we face.

1.1 Changing Attitudes toward Risk

- The shift from purely personal to organizational or public responsibility for risk is a recent phenomenon.

- Today's complex, technological society requires that the general public places its trust in "expert knowledge" [30].
  - Accordingly, responsibility for detection of and protection from hazards has been transferred from the public to the state, corporate management, engineers, safety experts, and other professionals.

- Public involvement has led to
  - Government regulation and
  - Creation of public interest groups to control hazards that were previously tolerated [113,172].

- Increased regional and national concern about safety has expanded in the past decade to include international issues.

1.2 Is Increased Concern Justified?

- Determining whether technological risk is increasing or not
  - Depends on the data used and its interpretation.

- Clearly, we will not find the answer to our question in such ambiguous and contradictory statistical data.

- In fact, the major changes occurring in the post–World War II era make most long-term historical risk data inapplicable to today's world.
  - Past experience does not allow us to predict the future when the risk factors in the present and future differ from those in the past.
  - But, examining these changes will help us understand the problems we face.

1.3 Unique Risk Factors in Industrialized Society

- **Risk** is a combination of
  - the ***likelihood*** of an accident and
  - the ***severity*** of the potential consequences.

- Different factors may affect these two components of risk:
  1.3.1 The Appearance of New Hazards
  1.3.2 Increasing Complexity
  1.3.3 Increasing Exposure
  1.3.4 Increasing Amounts of Energy
  1.3.5 Increasing Automation of Manual Operations
  1.3.6 Increasing Centralization and Scale
  1.3.7 Increasing Pace of Technological Change

1.4 How Safe is Safe Enough?

- The goal of system safety is to understand and manage risk in order to eliminate accidents or to reduce their consequences.
  - Anticipate accidents and their causes in before-the-fact hazard analyses
  - Eliminate or control hazards as much as possible throughout the life of a system.

- Unfortunately, hazards will never be eliminated completely from all systems.
  - Technical difficulty of anticipating and reducing risks
  - Design tradeoffs between safety, performance, and other goals

- Wolf suggests that [357]
  - "We may be in the presence of a contradiction of technological culture that can neither prevent potentially disastrous accidents nor accept their consequences."

- If this assumption is correct,
  - then the process for determining exactly which systems to build and what new technology to introduce into them becomes critical.
  - Anti-technology
    - Concludes that advanced technology should not be in dangerous systems.
  - Pro-technology
    - All new technology is good.
- Risk-benefit analysis
  - The only reasonable way to make technology and risk decisions
  - Often accepted as the only way to make technology and risk decisions, without realizing that there are alternatives.

1.4.1 Risk-Benefit Analysis and the Alternatives

- Often viewed as only way to make technology and risk analysis.
  - To utilitarian, catastrophic accidents are one of the risks of high technology, which in the long run, are outweighed by the technology's overall benefits.

- To apply this approach, we must be able to
  - Measure risk
  - Choose appropriate level for risk
  - Unfortunately, systems are usually designed and built while knowledge of their risk is incomplete or nonexistent.

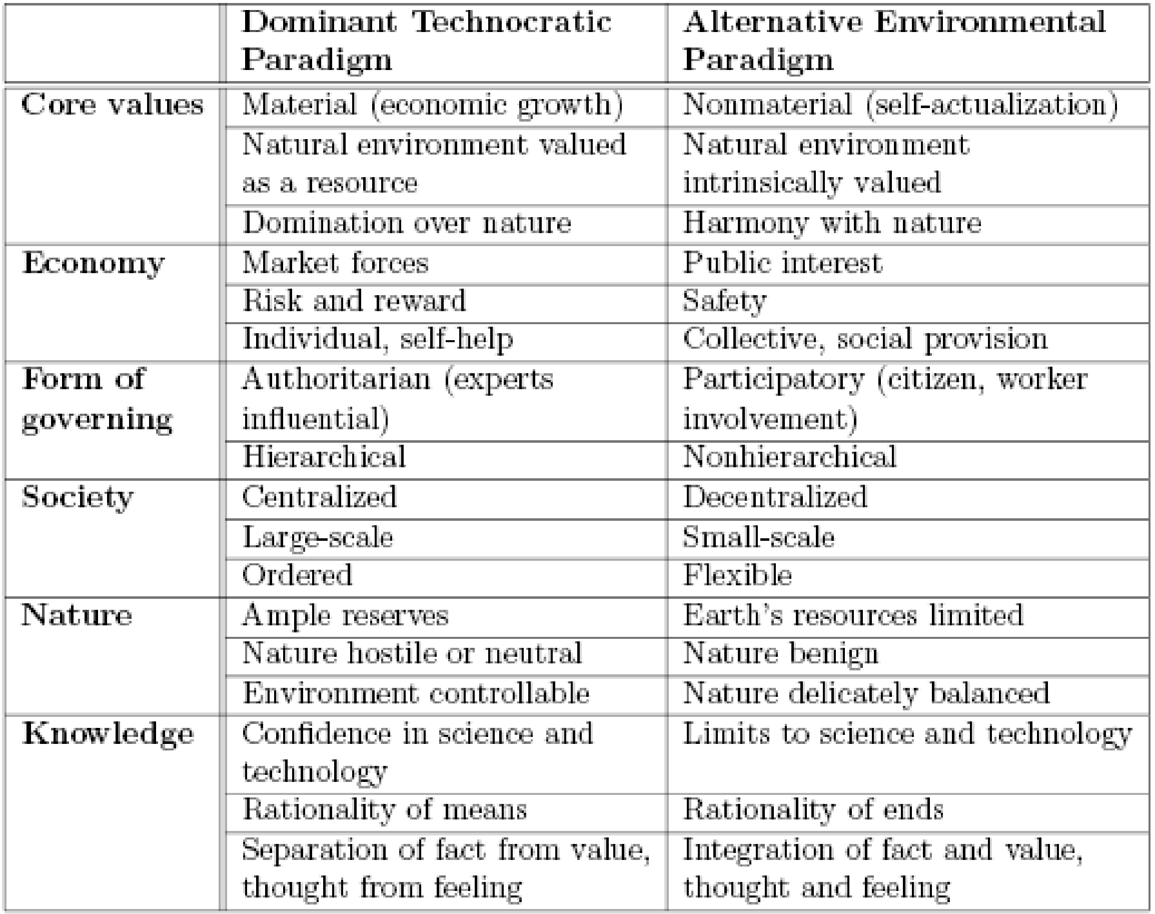
- Risk assessment
  - Impossible to measure before system is built.
  - Hard to perform and to determine *acceptable risk*
    - A threshold level selected below which risk will be tolerated

- Alternatives to the use of *acceptable risk*
  - *Optimal risk*
    - Involves a tradeoff that minimizes the sum of all undesirable consequences [100].
  - *Normal accidents* [259] by Perrow
    - Low catastrophic potential
      - Should be tolerated and could be further improved with modest effort.
      - Chemical plants, aircraft, air traffic control, dams, mining, and automobile
    - Moderate Low catastrophic potential
      - Should be strictly regulated.
      - Marine transport, recombinant DNA
    - High Low catastrophic potential
      - Should be abandoned and replaced.
      - Nuclear weapon, nuclear power

- There are no entirely satisfactory methods for making these decisions.
  - The decisions involve deep philosophical and moral questions - not simply technical choices.

1.4.2 Trans-Scientific Questions

- A basic problem with utilitarian approaches is that
  - They attempt to use scientific methods and arguments to answer what are fundamentally not scientific questions.
  - "Many of the issues that lie between science and politics involve questions that can be state in scientific terms but that are in principle beyond the proficiency of science to answer … I proposes the term 'trans-scientific' for such questions … Though they are, epistemologically speaking, questions of fact and can be stated in the language of science, they are unanswerable by science; they transcend science … In the current attempts to weigh the benefits of technology against its risk, the protagonists often ask for the impossible: scientific answers to questions that are trans-scientific [351]."



	Dominant Technocratic Paradigm	Alternative Environmental Paradigm
Core values	Material (economic growth)	Nonmaterial (self-actualization)
	Natural environment valued as a resource	Natural environment intrinsically valued
	Domination over nature	Harmony with nature
Economy	Market forces	Public interest
	Risk and reward	Safety
	Individual, self-help	Collective, social provision
Form of governing	Authoritarian (experts influential)	Participatory (citizen, worker involvement)
	Hierarchical	Nonhierarchical
Society	Centralized	Decentralized
	Large-scale	Small-scale
	Ordered	Flexible
Nature	Ample reserves	Earth's resources limited
	Nature hostile or neutral	Nature benign
	Environment controllable	Nature delicately balanced
Knowledge	Confidence in science and technology	Limits to science and technology
	Rationality of means	Rationality of ends
	Separation of fact from value, thought from feeling	Integration of fact and value, thought and feeling

Table 1.1: Alternative social paradigms (Adapted from Stephen Cotgrove. Risk, value conflict, and political legitimacy. In Richard F. Griffiths, editor, *Dealing with Risk: The Planning, Management, and Acceptability of Technical Risk*, Manchester University Press, Manchester, England, 1981, page 129.)

Chapter 2

COMPUTERS AND RISK

Introduction

We seem not to trust one another as much as would be desirable. In lieu of trusting each other, are we putting too much trust in our technology? ... Perhaps we are not educating our children sufficiently well to understand the reasonable uses and limits of technology.

- T.B. Sheridan
Trustworthiness of Command and Control Systems

And they looked upon the software, and saw that it was good. But they just had to add this one other feature. . .

- G.F. McCormick
When Reach Exceeds Grasp

Introduction

- We have a general-purpose machine.
  - If changes are needed, the instructions can be changed instead of building a different physical machine from scratch.

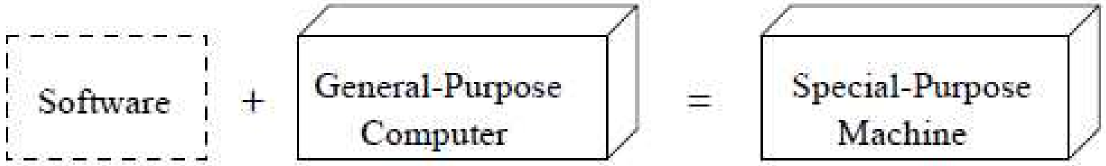


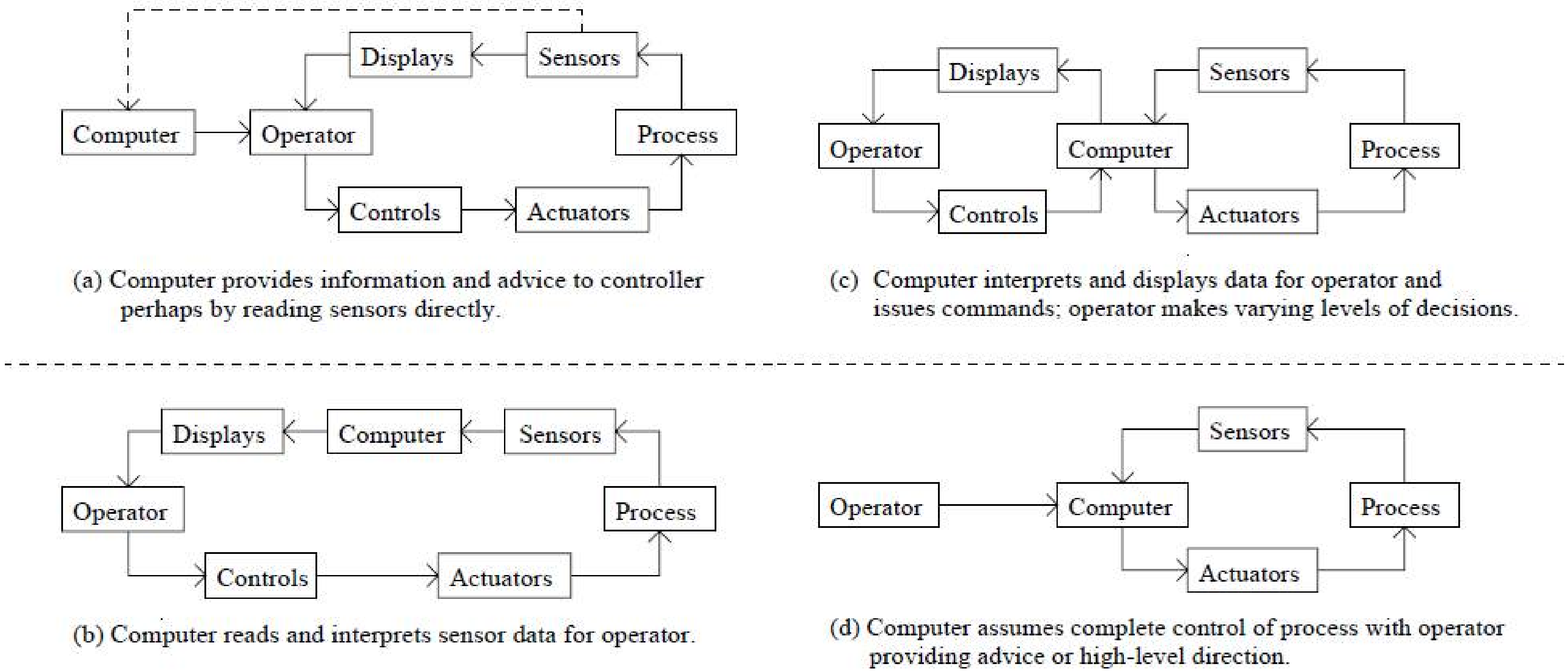
Figure 2.1: Software plus a general purpose computer creates a new special purpose machine.

- The advantages of computers have led to an explosive increase in their use, including their introduction into potentially dangerous systems.

2.1 The Role of Computers in Accidents

- Computers now control most safety-critical devices.
  - Often replace traditional hardware safety interlocks and protection systems.

- Computers can be used in safety-critical loops in several ways.(Figure 2.2)
  - Providing information or advice to a human controller upon request (2.2a).
  - Interpreting data and displaying it to the controller who makes the control decisions (2.2b).
  - Issuing commands directly, but with a human monitor of the computer's actions providing varying levels of input (2.2c).
  - Eliminating the human from the control loop completely (2.2d).



(a) Computer provides information and advice to controller perhaps by reading sensors directly.

(c) Computer interprets and displays data for operator and issues commands; operator makes varying levels of decisions.

(b) Computer reads and interprets sensor data for operator.

(d) Computer assumes complete control of process with operator providing advice or high-level direction.

Figure 2.2: Alternative uses of computers in control loops.

- The safety implications of computers exercising direct control over potentially dangerous processes are obvious (Figure 2.3a).

- Less obvious are the dangers when (as depicted in Figure 2.3b)
  - Software-generated data is used to make safety-critical decisions. (such as air traffic control and medical blood analyzers)
  - Software is used in design analysis (such as CAD/CAM).
  - Safety-critical data (such as blood bank data) is stored in computer databases.

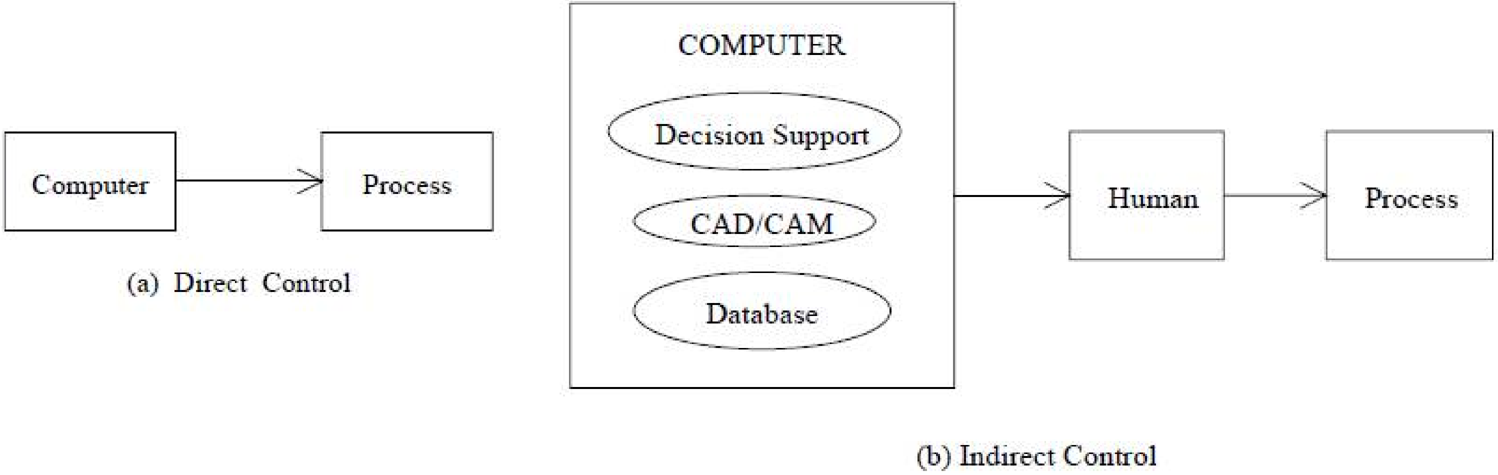


Figure 2.3: Direct and indirect control.

- In almost all cases, risk from incorrect computer operation is
  - Not eliminated by having a human in the loop.
  - Not even be reduced.

- Often the argument that software providing information or advice to humans is not safety critical
  - Used to avoid the difficult task of ensuring the safety of the software.
  - If system safety truly is to be increased, then all the components whose operation can directly or indirectly affect safety must be considered, and the related hazards must be eliminated or reduced.

- Computers also add difficulty and cost to accident investigations.
  - Few software engineering techniques have been developed to cope with safety problems.
  - Communication problems are making efforts to rectify these deficiencies more difficult. (system vs. software)

2.2 Software Myths

- If there are problems, why are computers being used so widely?
  - The basic reason is that computers provide a level of power, speed, and control not otherwise possible; they are also relatively light and small.

- Many others supposed that advantages of using computers are myths.

- The myths
  1. *The cost of computers is lower than that of analog or electromechanical devices.*
  2. *Software is easy to change.*
  3. *Computers provide greater reliability than the devices they replace.*
  4. *Increasing software reliability will increase safety.*
  5. *Testing software or "proving" (using formal verification techniques) software correct can remove all the errors.*
  6. *Reusing software increases safety.*
  7. *Computers reduce risk over mechanical systems.*

- In summary,
  - Computers have the potential to increase safety
  - This potential will be realized in the future.

- Computers will not go away.
  - Their use and importance in complex systems is only going to increase.

- To achieve and ensure safety in these systems,
  - Software must be included in the system-safety activities, and
  - The software must be specifically developed to be safe using the results of ***system hazard analysis***.

KONKUK UNIVERSITY

2.3 Why Software Engineering is Difficult ?

- Parnas [253] and Shore [314] have written excellent descriptions of the unique engineering problems in constructing complex software.
  - Analog versus Discrete State Systems
  - The “Curse of Flexibility”
  - Complexity and Invisible Interfaces
  - Lack of Historical Usage Information

- Analog versus Discrete State Systems
  - In control systems, the computer is usually simulating the behavior of an analog controller.
  - Translation of the function from analog to digital form may introduce inaccuracies and complications.
    - Continuous functions can be difficult to translate to discrete functions.
    - Discrete functions may be much more complex to specify.
  - Factors such as time, finite-precision arithmetic, and concurrency are difficult to handle.
  - Physical continuity in analog systems also makes them easier to test than software.

- The “Curse of Flexibility”
  - The ease of change encourages major and frequent change, which often increases complexity rapidly and introduces errors.
  - Flexibility also encourages the redefinition of tasks late in the development process in order to overcome deficiencies found in the other parts of the system.
  - Software has no corresponding physical limitations or natural laws.
    - Makes it too easy to build enormously complex designs. (Figure 2.4)

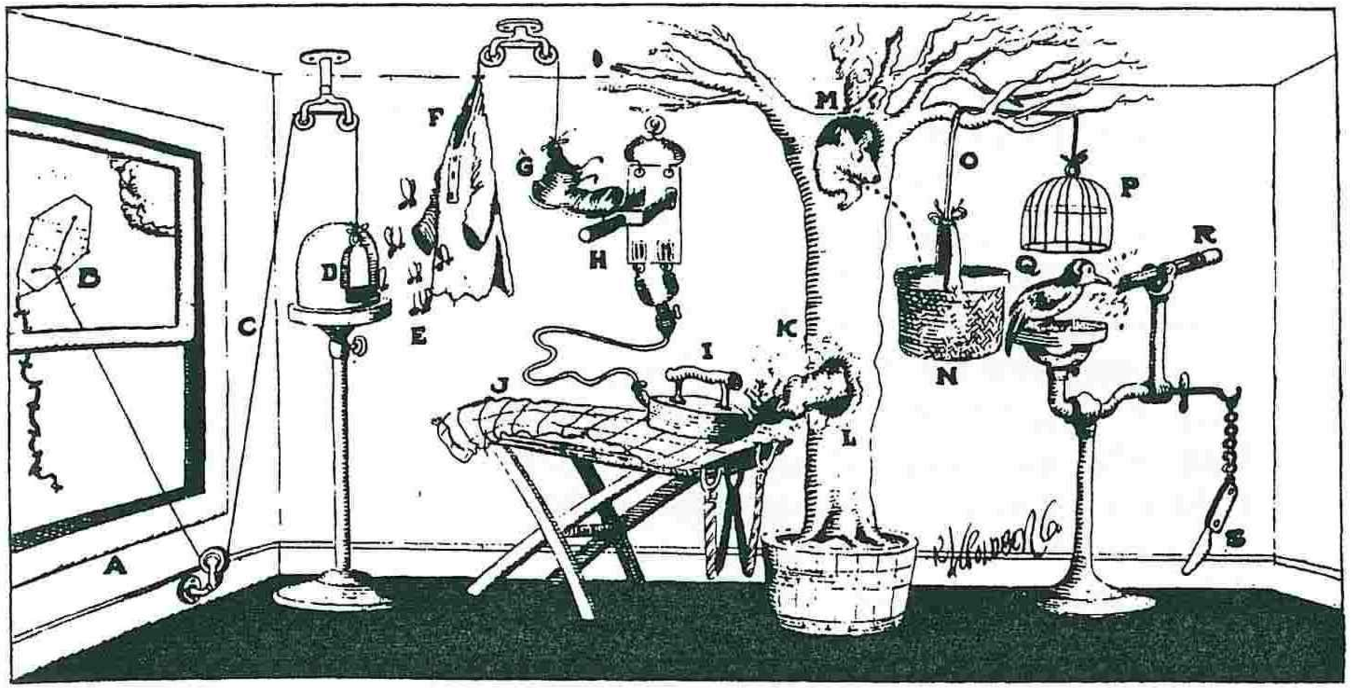


FIGURE 2.4

The simplified pencil sharpener. Open window (A) and fly kite (B). String (C) lifts small door (D), allowing moths (E) to escape and eat red flannel shirt (F). As weight of shirt lessens, shoe (G) steps on switch (H), which heats electric iron (I) and burns hole in pants (J). Smoke (K) enters hole in tree (L), smoking out opossum (M) which jumps into basket (N), pulling rope (O) and lifting cage (P), allowing woodpecker (Q) to chew wood from pencil (R), exposing lead. Emergency knife (S) is always handy in case opossum or the woodpecker gets sick and can't work. (From *Rube Goldberg vs. The Machine Age* by Rube Goldberg © 1968 King Features. Edited by Clark Kincaid, New York: Hastings House Publishers. Reprinted with special permission of King Features Syndicate.)

- Complexity and Invisible Interfaces
  - One way to deal with complexity is to break the complex object into pieces or modules.
    - However, the large number of interfaces created introduce uncontrollable complexity into design.
  - Finding good software structure has proven to be surprisingly difficult [253].
  - Software has no physical connections.
    - Logical connections are cheap and easy to introduce.
    - Complex interfaces are as easy to construct as simple ones, perhaps easier.
  - Moreover, the interfaces between software components are often “invisible” or not obvious.
    - It is easy to make anything depend on anything else.

- Lack of Historical Usage Information

  - No historical usage information is available to allow measurement, evaluation, and improvement on standard designs.
    - Software is almost always specially constructed.
    - Physical systems benefit from the experience gained by the use of standard designs over long periods of time and in varied environments.

2.4 The Reality We Face

- When systems were composed only of electromechanical and human components,
  - Engineers knew that random, wear-out failures and human errors could be reduced and mitigated but never completely eliminated.
  - On the other hand, design errors, could be handled fairly well through testing and reuse of proven designs.

- Software has only design errors.
  - The primary approach used to deal with reliability and safety problems has been simply to get the software correct.

  - Theoretically, the possibility does exist for finding a set of techniques or methodology that will allow us to build perfect software.
  - In reality, the time to create perfect software is never there, and perhaps it never can be.

Chapter 3

A HIERARCHICAL VIEW OF ACCIDENTS

Introduction

> *We saw that NASA had no system for fixing the [Shuttle O-ring] problem, even though engineers were writing letters like, "HELP!" and 'This is a RED ALERT!" Nothing was done.*
>
> *- Richard P. Feynman*
> *An Outsider's Inside View of the Challenger Inquiry*

- In order to devise effective ways to prevent accidents, we must first understand what causes them.

- Determining the cause of an accident is much more complex than is often imagined.

- Before looking at what causes accidents, we need to clarify just what “cause” means.

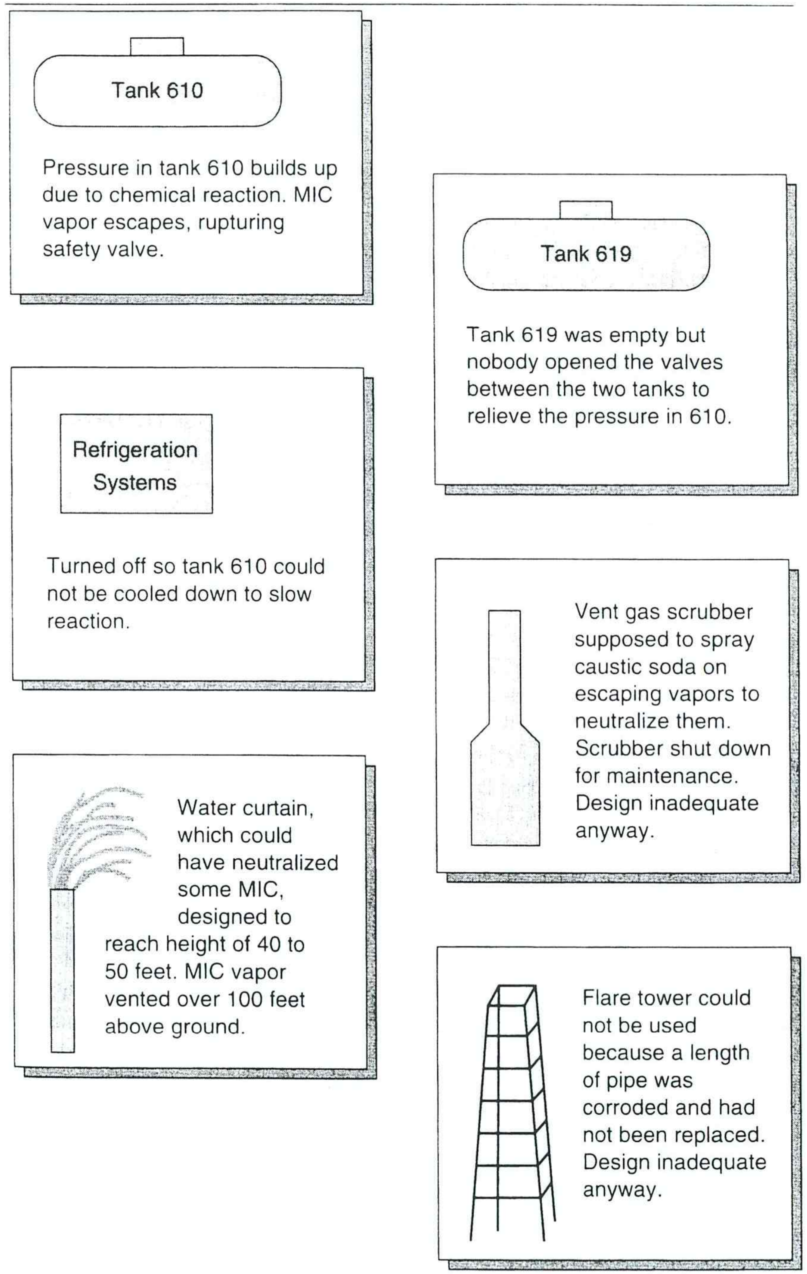
3.1 The Concept of Causality

- The complexity of conditions and events contributing to an accident is not unusual.
  - “The vessel *Baltic Star*, registered in Panama, ran aground at full speed on the shore of an island in the Stockholm waters on account of thick fog. One of the boilers had broken down, the steering system reacted only slowly, the compass was maladjusted, the captain had gone down into the ship telephone, the outlook man on the prow took a coffee break and the pilot had given an erroneous order in English to the sailor who was tending the rudder. The latter was hard of hearing and understand only Greek.”

- Some accidents are so complex and so permeated with uncertainties.
  - They defy any simple explanation of their cause or even an exhaustive listing of all the factors involved.
  - The accidental release of methyl isocyanate (MIC) at Bhopal, India, 1984

- Many causes may result in the accident.
  - It is not uncommon for a company to turn off passive safety devices to save money.
  - A number of thresholds established for the production of and exposure to MIC were routinely exceeded.
  - No knowledge or training about how to handle non-routine events.

- Any of these perspectives or "causes" is inadequate by itself to understand the accident and to prevent future accidents.



Pressure in tank 610 builds up due to chemical reaction. MIC vapor escapes, rupturing safety valve.

Tank 619 was empty but nobody opened the valves between the two tanks to relieve the pressure in 610.

Turned off so tank 610 could not be cooled down to slow reaction.

Vent gas scrubber supposed to spray caustic soda on escaping vapors to neutralize them. Scrubber shut down for maintenance. Design inadequate anyway.

Water curtain, which could have neutralized some MIC, designed to reach height of 40 to 50 feet. MIC vapor vented over 100 feet above ground.

Flare tower could not be used because a length of pipe was corroded and had not been replaced. Design inadequate anyway.

FIGURE 3.1
Some causal factors at Bhopal.

- The accident illustrates some of the complexity in assigning causes to accidents.
  - The distinction between sufficient and necessary is important [190].

- John Stuart Mill (1806~1873) defined a *cause* as a set of sufficient conditions.
  - “The cause is the sum total of the conditions, positive and negative, taken together, the whole of the contingencies of every description, which being realized, the consequence invariably follows. [222]”

- In this book, a cause of an event (accident)
  - Composed of a set of conditions, each of which is necessary and which together are sufficient for the event to occur.
  - The individual conditions will be call *causal* or *hazardous conditions* or *factors* to distinguish them from causes.

3.2 Subjectivity in Ascribing Causality

- Each person may attribute an accident to a different cause.
  - Leplat concluded after examining the research data [175],
    - "Causal attribution depends on some characteristics of the victim and of the analyst (hierarchical status, degree of involvement, and job satisfaction) as well as on the relationships between the victim and the analyst and on the severity of the accident."
  - Causal identification may also be influenced by the data collection methods.
- The casual factors selected for a particular accident may involve a great deal of subjectivity and need to be interpreted with care.

3.3 Oversimplification in Determining Causality

- Oversimplification
  - A condition is selected as the cause, because it is the last condition to be fulfilled before the effect takes place.
  - Regarding its precondition as the most conspicuous.

- Oversimplification of the causal factors in accidents can be hindrance in preventing future accidents.

- Some common types of oversimplifications include
  - 3.3.1 The Legal Approach to Causality
  - 3.3.2 Human Error
  - 3.3.3 Technical Failures
  - 3.3.4 Organizational Factors
  - 3.3.5 Multifactorial Explanations of Accidents

3.3.1 The Legal Approach to Causality

- Lawyers and insurers often
  - Oversimplify the causes of accidents.
  - Identify what they call the *proximate* (immediate or direct) cause.
  - Identify a principal factor for practical and liability reasons.
- The goal is to determine which parties in a dispute have the legal liability to pay damages.
  - It is of little use if the goal is to understand and prevent accidents.
- It may be a hindrance because the most relevant factors may be ignored for nontechnical reasons.

3.3.2 Human Error

- The most common oversimplification in accident reports is blaming the operator.
  - Virtually any accident can be ascribed to human error in this way.
- However, considering that alone as a cause and the elimination of humans or of human errors as the solution is
  - Too limiting to be useful in identifying what to change in order to increase safety most effectively.
- An understanding of the role of humans in accidents is so important in reducing risk.
  - It will be discussed in depth Chapters 5 and 6.

3.3.3 Technical Failures

- Oversimplification is concentrating only on technical failures and immediate physical events.
  - May lead to ignoring some of the most important factors in an accident.
  - An explosion at a chemical plant in Flixborough, England, 1974.
  - The British Court of Inquiry concluded that
    - "There were undoubtedly certain shortcomings in the day to day operations of safety procedures, but none had the least bearing on the disaster or its consequences and we do not take time with them."

3.3.4 Organizational Factors

- Large-scale engineering systems are more than just a collection of technological artifacts.
  - A reflection of the structure, management, procedures, and culture of the engineering organization that create them
  - A reflection of the society in which they were created

- The causes are frequently rooted in the organization .
  - Culture, management, and structure
  - They are all critical to the eventual safety of the engineering system.
    - The Three Mile Island (TMI) accident
    - The Space Shuttle Challenger accident

- In most of the major accidents of the past 25 years, technical information on how to prevent the accident was known and often even implemented.
  - But in each case, the technical information and solutions were negated by organizational or managerial flaws.

3.3.5 Mutifactorial Explanations of Accidents

- Accidents or hazards will depend on
  - A multiplicity of causal factors and
  - Each event on a complex combination of conditions,
  - including technical, operator, and organizational or societal conditions and actions.

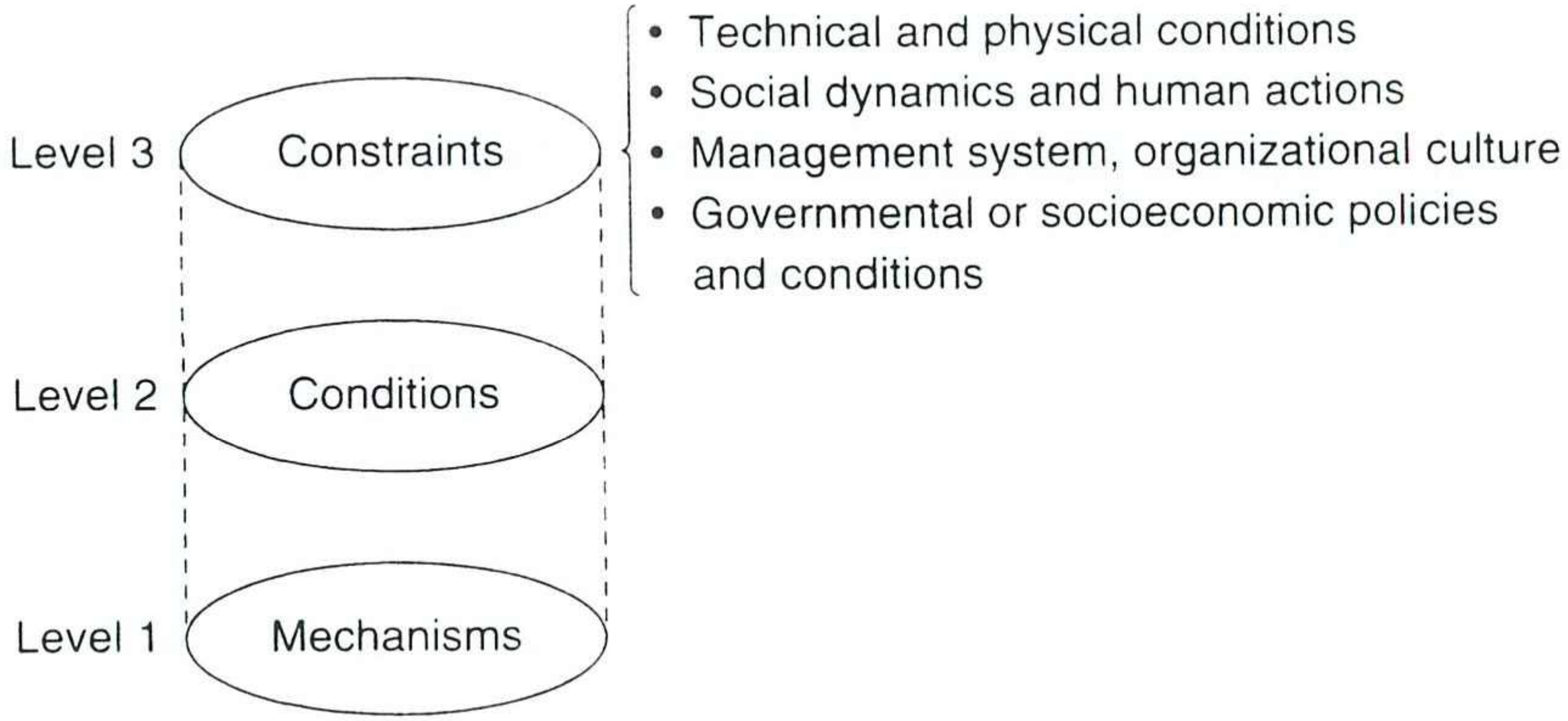
- Therefore, consideration of the conditions leading to accidents will be more useful than simplistic explanations.

3.4 A Hierarchical Approach to Causality

- Specifying all necessary conditions may be impractical.

- If the goal is to prevent future accident,
  - The investigation must be carried out on multiple hierarchical levels including technical, human, organizational, and regulatory perspectives.

- Three-level model of understanding accidents by Lewycky [190]
  - Level 3
    - "root" causes of the accident
  - Level 2
    - Conditions or lack of conditions that allowed the events at the first level to occur.
  - Level 1
    - The mechanism of the accident – the chain of events



Level 3 Constraints

Level 2 Conditions

Level 1 Mechanisms

- Technical and physical conditions
- Social dynamics and human actions
- Management system, organizational culture
- Governmental or socioeconomic policies and conditions

FIGURE 3.2
Hierarchical model of accident causes.

- Patches to particular parts of the system may be ineffective in preventing future accidents.
  - Accidents rarely repeat themselves in exactly the same time.
    - Challenger
    - DC-10 cargo-door saga

- The lessen is clear.
  - To reduce the risk of accidents significantly, root causes must be identified and eliminated.
  - If we are unable to eliminate a particular causal factor, we may still be able to provide protection against it leading to an accident.

Chapter 4

ROOT CAUSES OF ACCIDENTS

Introduction

> *The [FAA] administrator was interviewed for a documentary film on the [Paris DC-10] accident. He was asked how he could still consider the infamous baggage door safe, given the door failure proven in the Paris accident and the precursor accident at Windsor, Ontario. The Administrator replied - and not facetiously either - ‘Of course it is safe, we certified it.”*
>
> *- C.O. Miller*
> *A Comparison of Military and Civilian Approaches to Aviation Safety*

- The root causes (or level-three causes) of accidents can be divided into three categories:
  (1) Deficiencies in the safety culture of the industry or organization
  (2) Flawed organizational structures
  (3) Superficial or ineffective technical activities

4.1 Flaws in the Safety Culture

- Safety Culture
  - General attitude and approach to safety
  - Reflected by those who participate in an industry or organization: management, workers, and government regulators

- Major accidents often stem from flaws in this culture, especially
  (1) Overconfidence and complacency
  (2) A disregard or low priority for safety
  (3) Flawed resolution of conflicting goals

4.1.1 Overconfidence and Complacency

- Complacency may be one of the most important factors in risk.
  - Nuclear energy
    - An NRC post-TMI Lessons Learned Task Force [3]
      - "The mindset regarding serious accidents was probably the single most important human factor with which this industry and the NRC has to content."
    - Only a month before the Chernobyl accident, the British Secretary of State for Energy repeated the often stated belief that "the nuclear energy is the safest form of energy yet known to man." [260]
  - Avionics
    - The report of the Presidential Commission on the Space Shuttle Challenger Accident [295] pinpointed two related causes as
      - Complacency
      - Belief that "Less safety, reliability, and quality-assurance activity is required during 'routine' Shuttle operations."
  - Chemical plants
    - In the Bhopal accident, the work mangers said after informed of the accident,
      - "The gas leak just can't be from my plant. The plant is shut down. Our technology just can't go wrong, we just can't have leaks." [30]

- Various aspects of complacency
  - Discounting risk
  - Over-relying on redundancy
  - Unrealistic risk assessment
  - Ignoring high-consequence, low-probability events
  - Assuming risk decreases over time
  - Underestimating software-related risks
  - Ignoring warning signs

- Discounting Risk
  - When people attempt to predict the system risk, they explicitly or implicitly multiply events with low probability - assuming independency - and come out with impossibly small numbers, when, in fact, the events are dependent.
  - Titanic coincidence [202]
    - Major accidents are often preceded by a belief that "They cannot happen."
    - The effect says that the magnitude of disasters decreases to the extent that people believe that disasters are possible and plan to prevent them or to minimize their effects.

- Over-relying on Redundancy
  - Redundancy and diversity are often used in an attempt to avoid failures and increase reliability.
    - Diversity in a chemical plant protection system
  - “Common-cause failures” are failures that are ‘coincident’ in time.
    - Because of dependencies among conditions leading to the events
    - Many accidents can be traced to common-cause failures in redundant or diverse systems.
  - Examples
    - In Browns Ferry nuclear Power Plants in Alabama, a fire burned uncontrolled for seven and a half hours in March 1975.
    - In Challenger flight, the failure of the primary O-ring caused conditions that led to the failure of the secondary O-ring.
    - U.S Strategic Air Command’s (SAC’s) special communication lines failures
    - At Bhopal, a number of independent safety devices ‘failed’ at the same time.

- Unrealistic Risk Assessment
  - Unrealistic risk assessments may also lead to complacency and lack of appropriate safety activities.
  - In case of Therac-25,
    - They responded that the probabilistic risk assessment showed that accidents were impossible, and no action was taken.
    - A hazard analysis had been performed on the Therac-25, but software errors were excluded. [187]
    - Next time, they included software in the fault tree, but assigned a probability of 10^{-4} to every types of software errors. But there is no justification for such a number.

- Probabilistic safety analysis can be useful in making decisions about the acceptability of a particular design.

- However, numerical assessments measure only what they can measure and not necessarily what needs to be measured.
  - Immeasurable factors such as design flaws and management errors are ignored.
  - The underlying assumptions of the assessment (such as certain failures are independent) are often ignored.

  - They frequently change the emphasis in development from making a system safer to proving that the system is safe as designed.
  - Designers may work to achieve the probabilistic goal only and proceed no further, even where additional corrective action is possible.

- Therefore, quantitative analysis should be interpreted and used with care.
  - "The numerical game in risk assessment should only be played in private between consenting adults as it is too easy to be misinterpreted." [306]

- Ignoring High-Consequence, Low-Probability Events
  - The most likely hazards are usually controlled.
  - But, hazards with high severity and (assumed) low possibility are dismissed as not worth investing resources to prevent.
    - A common discovery after accident is that the events involved were recognized before the accident, but were dismissed as incredible.
  - Independence and other assumptions used in either informal or formal risk assessments should be scrutinized with great skepticism.
    - The potential for catastrophic accidents of a type not yet experienced should not be casually dismissed.

- Assuming Risk Decreases over Time
  - The belief that a system must be safe because it has operated without an accident for many years.
    - Therac-25
    - Industrial robots
    - Nitro methane
  - Carrying out an operation in a particular way for many years does not guarantee that an accident cannot occur.
    - In reality, risk may decrease, remain constant, or even increase over time.

- Underestimating Software-Related Risks
  - A new form of complacency appears.
    - With the introduction of computers into the control of complex systems
    - “Software cannot fail and all errors will be removed by testing.”
  - Software design errors are much harder to find and eliminate than hardware.
    - Hardware failure modes are generally much more limited, and therefore it is usually easier to build software in protection against them.
  - Underestimation of software risks can also be seen in some of the new software standards for civil aircraft and nuclear power plants.
    - Software functions are defined to be safety critical only if they alone lead to an accident.

- Ignoring Warning Signs
  - Accidents are frequently preceded by public warnings or by a series of minor occurrences or other signs.
    - Often these precursors are ignored, because individuals do not believe that a major accident is possible.
    - Many cases
  - Because people tend to underestimate risks and ignore warnings,
    - One or more serious accidents are often required before action is taken.
  - Using this information to attempt to predict and prevent accidents before they occur seems reasonable.

4.1.2 Low Priority Assigned to Safety

- Efforts for safety will most certainly be ineffective without support from top management.
  - The entire organization must have a high level of commitment to safety.
    - Employee must believe that they will be supported by the company if they reasonably choose safety over other goals.
    - The informal rules (social processes) as well as the formal rules of the organizational culture must support the overall safety policy.
  - The lead must come from the top and permeate every organizational level.
    - Management's most important actions in preventing accidents involve selecting and implementing organizational priorities.
      - Many managers recognize that safety is a good business over the long term; others put short-term goals ahead of safety.
      - In general, user groups are much more effective than government regulatory agencies in building management support for safety activities.

4.1.3 Flawed Resolution of Conflicting Goals

- Safety needs to be recognized as
  - Not only a high priority goal,
  - But also procedures for resolving goal conflicts.
- Safety often suffers in tradeoffs between conflicting goals because of a dearth of hard data on benefits.
  - Cost and Schedule vs. Safety
  - Sacrificing a sure productivity gain in favor of preventing a seemingly low-probability accident may not seem like a reasonable course of action.
- A belief that "safety conflicts with the basic benefits of technology" is common.

- The belief is also a factor in the use of downstream protection rather than upstream hazard elimination and control.
  - Downstream approach: adding protection features to a completed design
  - Upstream approach: designing safety into the system from the beginning by eliminating or reducing hazards in the original design

- Safety can be increased using upstream methods without giving up any of the benefits of the design.

- Downstream approach may eliminate the need for some types of design tradeoffs, but
  - It may increase the overall cost of the system.
  - Schedule delays also increases.

- Upstream approach may result in
  - (by Eliminating or controlling hazards in the early design stages and making tradeoffs early)
  - Lower costs during both development and the overall system lifetime
  - Fewer delays and less need for costly redesign
  - Lower risk.

- The difficulty in any discussion of cost and tradeoffs with respect to safety is
  - Benefits from system safety programs show up only in the long run and observable only indirectly.

- However, there is evidence that system safety programs can save money.
  - It often costs no more to build an inherently safe system, if the system is designed correctly in the first place.
  - An inherent safe system is often cheaper than one with a lot of overdesign in the form of added-on protection devices.
  - Designing in safety from the start is much cheaper than making retroactive design changes.
  - Those costs may be minimal in comparison with the potential costs of a major accident.

- It is widely believed that increasing safety slows operations and decreases performance.
  - In reality, experience has shown that, over a sustained period, a safer operation is generally more efficient and can be accomplished more rapidly.
  - One reason is that stoppage and delays are eliminated. [108]

- The belief that "safer systems cost more" or "building safety in from the beginning necessarily requires unacceptable compromises with other goals" is not justified.

4.2 Ineffective Organizational Structure

- Many accident investigations uncover a sincere concern for safety in the organization.
  - However, they found organizational structures in place that were ineffective in implementing that concerns.
  - Zebroski [360] examined 4 major accidents and found similar deficiencies in organizational style and structure.
    - Bhopal, Challenger, Three Mile Island, and Chernobyl

- Basic management principles should apply to safety as well as to other quality goals.

- Nevertheless, some specific issues in organizational structure appear to be related to accidents:

  (1) Diffusion of Responsibility and Authority

  (2) Lack of Independence and Low-Level Status of Safety Personnel

  (3) Limited Communication Channels and Poor Information Flow

4.2.1 Diffusion of Responsibility and Authority

- Accidents are often associated with
  - A fractured, organizationally dispersed safety staff
  - Ill-defined responsibility and authority for safety matter.

- Problems arise especially when responsibility is divided across organizational boundaries.

- Some person or group required to
  - Integrate the information and make sure it is available to all decision makers.
  - Because different contractors all makes decisions influencing safety.

- A large organizational distance between decision makers and those with technical awareness and competence is a common problem in engineering organizations.

4.2.2 Lack of Independence and Low-Level Status of Safety Personnel

- Location of safety organization in the management structure
  - Important factors in terms of both reporting responsibility and funding
  - Dependence from the project or program management for which it provides oversight or input is important.
    - Challenger
    - Flixborough explosion

- Serious accidents are often associated with a low status of the safety personnel and their lack of involvement in critical discussions and decision making.
  - The status of safety personnel and their involvement in decision making is a clear sign to everyone in the organization of the real emphasis that management places on safety.

4.2.3 Limited Communication Channels and Poor Information Flow

- Software development organization is often totally divorced from the safety organization.
  - Information flow is nonexistent or limited.
  - Serious problems can often be traced to such disconnects.

- Communication paths and information need to be explicitly defined.
  - Reference channel (downward)
  - Measuring channel (upward)

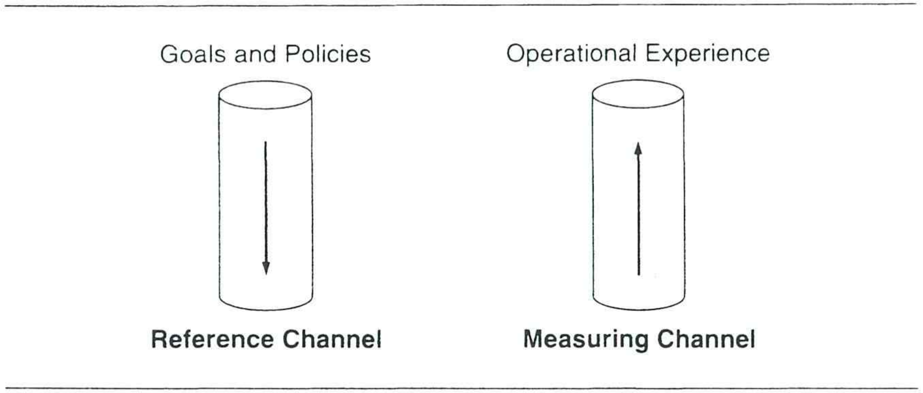


FIGURE 4.2
Information channels.

4.3 Ineffective Technical Activities

- Root causes of accidents often reflect poor implementation of the specific activities necessary to achieve an acceptable level of safety.
  (1) Superficial Safety Efforts
  (2) Ineffective Risk Control
  (3) Failures to Evaluate Changes
  (4) Information Deficiencies

4.3.1 Superficial Safety Efforts

- Sometimes safety organizations are caught in bureaucratic environment.
  - Paperwork may be completed to meet milestones, irrespective of quality.
  - Cosmetic system safety
    - Called by Child [51]
    - Characterized by superficial safety efforts and perfunctory bookkeeping

- In some cases,
  - Adequate safety analyses are performed,
  - But no follow-up is done to ensure that the hazards.
    - The Union Carbide plant in Bhopal

4.3.2 Ineffective Risk Control

- The majority of accidents are not the result of lack of knowledge about how to prevent them, but of failure to use that knowledge or to use it effectively.
  - The way to prevent particular hazards may not be known to the people concerned.
  - In some accidents, the hazards are identified and efforts are made to control them, but that control is inadequate.
  - In fact, technological safety fixes themselves sometimes create accidents.
    - A partial meltdown at the Detroit Fermi nuclear reactor
    - French weather balloons

- To make our technology fixes more effective, we need to understand why they sometimes do not work.

- Four major factors are:
  (1) Failing to eliminate basic design flaws
  (2) Safeguards based on false assumptions
  (3) Complexity
  (4) Using risk control devices to reduce safety margins

4.3.3 Failures to Evaluate Changes

- Accidents often involve a failure to reevaluate safety after changes are made.
  - Many examples.
- Any changes in either hardware or software must be evaluated to determine whether safety has been compromised.
  - Changes in critical hardware or software require complete regression testing.
  - They also require system and software safety analysis.
- One of the requirements for performing an effective change analysis is that the safety-related design decisions have been documented.
  - Need to specify the constraints and assumptions of the safety assessment and the safety-related design features, so the maintainers do not accidentally violate the assumptions or eliminate the safety features.
  - Safety design decisions must be documented and used when making decisions.

4.3.4 Information Deficiencies

- Feedback of operational experience
  - One of the most important sources of information in designing, maintaining, and improving safety.

- Two types of data are important, and should be both collected and used.
  - (1) Information about incidents and accidents in other systems
  - (2) Trend and incident data in the system being maintained

- To be useful for safety analysis, the criticality of software errors reported should be identified.
  - The information obtained about these errors should be used to improve the process or to make the software or system safer.

- Most problem with respect to safety is
  - The tendency to replace electromechanical systems with computer systems without incorporating the knowledge about accident prevention that has been passed down through standard hardware design and codes of practice.

4.4 Summary

- This chapter has looked at the level-three factors that have been implicated in past accidents.
  - Complacency is a common factor in major accidents.
  - Flaws in the safety culture related to problems in resolving conflicting goals.
  - In other cases, a sincere concern for safety exists, but the organizational structure was ineffective for implementing that concerns.
  - Implementation of the specific technical activities necessary to achieve an acceptable level of safety is poor.

- In later chapters, information about these crucial factors is used to generate and evaluate accident models and to design risk reduction programs.
  - But first, we need to examine the relationship between human and accidents in more depth.

Chapter 5

HUMAN ERROR AND RISK

Introduction

> *In childhood, falling down may be part of growing up. But do we need the falls of hundreds of "London Bridges," Tacoma Narrow bridges, and Hyatt walkways? De we need DC-10's to plummet, commercial and private aircraft to collide and devastate a neighborhood, nuclear plants to release radioactive fallout, and other such falls to wake up to the critical importance of human factors in design, analysis, production, installation, maintenance, training, and operation of our product?*
>
> *- Richard Hornick*
> *Dreams: Design and Destiny*

- Computers have made automation of many manual operations feasible.
  - Increased capacity and productivity
  - Reduction of manual workload and fatigue
  - Relief from and more precise handling of routine operation
  - Elimination of individual human differences

- Economics plays an important role in automation.
  - Savings in energy cost
  - But, all savings are potential and not necessarily completely realizable.
    - How one automates will determine the actual savings
    - In addition, automated equipment is not cheap.

- As system control requirements grow more complex, they may exceed human capabilities.
- Automation may increase safety.
  - Many accident are attributed to human error.

- Important issues for system designers to understand
  - Whether automation increases safety or not
  - The role of human errors in accidents

5.1 Do Humans Cause Most Accidents?

- A common assumption is that human operators cause the majority of accidents.
  - But, the real question is whether
    - They cause the accidents
      or
    - They are merely swept up in the events and blamed for them

- A common cited statistics is that
  - 85% of work accidents are due to unsafe acts by human rather that unsafe conditions [140].
  - 88% of all accidents are caused primarily by dangerous acts of individual workers [118].
  - 60 ~ 80% of accidents were the direct result of the operator’s loss of control of energies in the system [150].

- However, this data should not be merely taken at face value, but needs to be examined more closely.

- Are there other explanations that do not cast so much blame on the operator?
  - The data may be biased and incomplete.
  - Positive actions are usually not recorded.
  - Blame may be based on the premise that operators can overcome every emergency.
  - Operators often have to intervene at the limits.
  - Hindsight is always 20/20.
  - Separating operator errors from design errors is difficult and perhaps impossible.

- In summary, human actions both prevent and contribute to accidents in high technology systems.
  - Accidents or incidents occur because of
    - Human error
    - Machine failure
    - Poor design
    - Inadequate procedure
    - Social or managerial inadequacies
    - Interaction of all these factors
  - Assignment of blame to operator error alone ignores the multiple causality of accidents and the relationship between human actions and the context in which they occur.

- But, even if humans have been blamed incorrectly for accidents, they certainly have contributed to many of them.
  - Why we should not just eliminate humans from system and replace them with computers?

5.2 The Need for Humans in Automated Systems

- Computers and other automated devices are best at trivial and straightforward tasks.
  - An adequate response must be determined for every situation.
  - However, not all conditions are foreseeable.
    - Those that arose from a combination of events
    - Those that can be predicted are programmed by error-prone humans

- System behavior and performance become increasingly difficult to understand and anticipate.
  - Because system design becomes more complex and requires coordination and integration of various types of expertise.

- Designers may fail to
  - understand fully the system characteristics.
  - Anticipate all the environmental conditions under which the system must operate.

- Human operators are included in complex systems, because they are adaptable and flexible, unlike computers.
  - In a computer, the response to each situation should be preprogrammed.
  - Humans are able to look at tasks as a whole and to adapt both the goals and the methods used to achieve them.

- Human error is the inevitable side effect of this flexibility and adaptability.
  - Abound of examples that operators ignored prescribed procedures in order to solve a problem, but those instances when the decision was incorrect are more likely to get attention.

- Many incidents demonstrate the danger of trying to prescribe regulations, procedures, or algorithms.
  - Human error is often defined as behavior that is inconsistent with normal, programmed patterns.

5.3 Human Error as Human-Task Mismatch

- Rasmussen argues that 'human error' is not a useful term and should be replaced by considering such events to be 'human-task mistakes.'
  - The tasks in modern automated systems involve problem solving and decision making that in turn require
    - Adaptation
    - Experimentation
    - Optimization of procedures
  - He explains the relationship between the three behaviors and human error using three-level model based partly on GPS(General Theory of Problem Solving).
    - Skill-based behavior
      - Characterized by patterns of movement generated from a dynamic, internal model of the world
    - Rule-based behavior
      - Depends on implicit models of the environment embedded in procedural rules
    - Knowledge-based behavior
      - Depends on structural models of environment

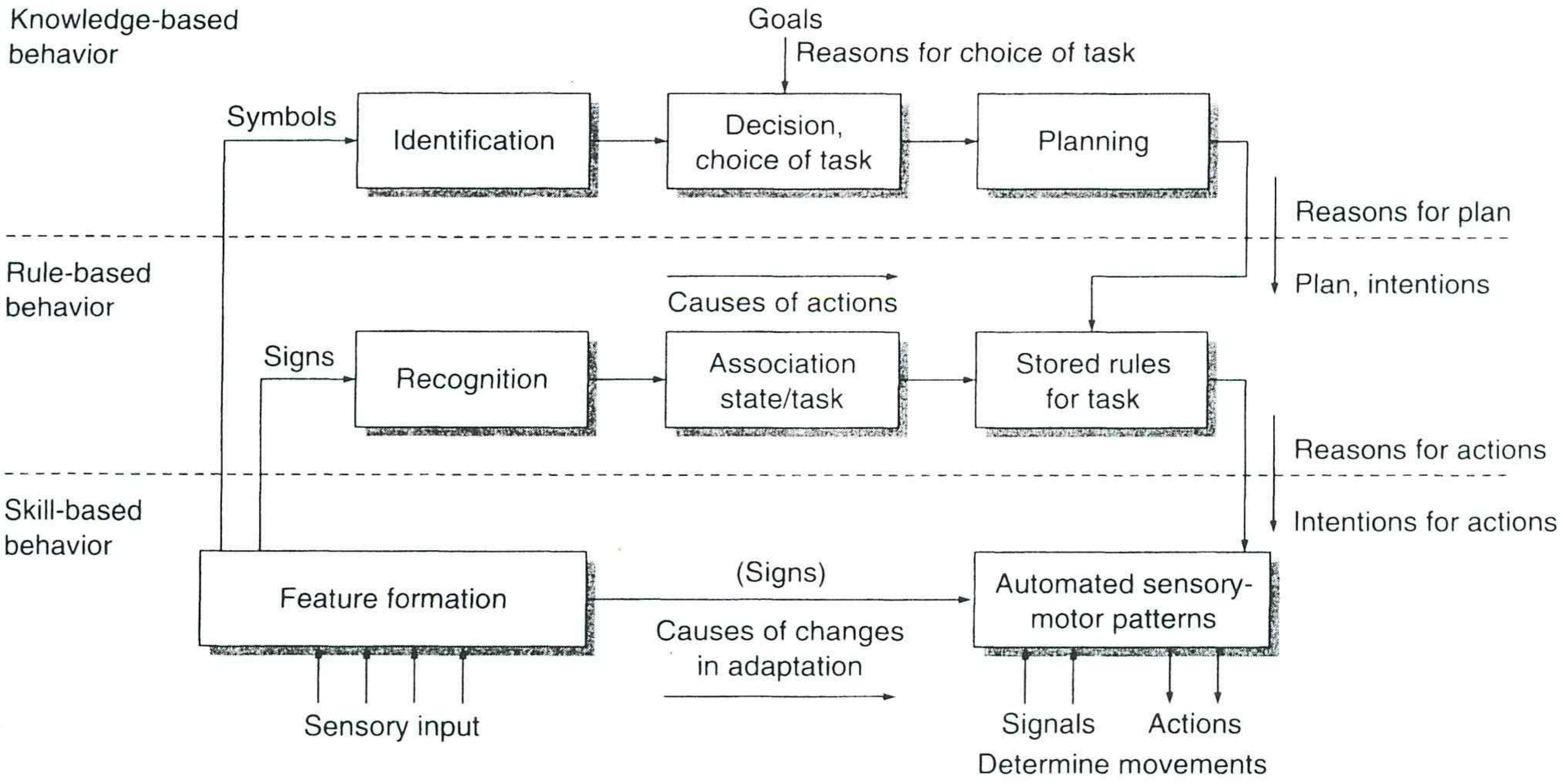


FIGURE 5.3
Rasmussen's model of cognitive control. (Source: Adapted from J. Rasmussen, "What Can Be Learned from Human Error Reports?" in K. Duncan et al., eds., *Changes in Working Life*. New York: John Wiley & Sons, 1980. Reprinted with permission.)

- Human errors can be considered to be unsuccessful experiments performed in an “unkind” environment,
  - Where an unkind work environment is one in which it is not possible for the human to correct the effects of inappropriate variations in performance before they lead to unacceptable consequences.

- Rasmussen concluded
  - Human performance is a balance between a desire to optimize skills and a willingness to accept the risk of exploratory acts.
  - Attempts to eliminate such behavior by admonishing humans to take more care or by enforcing predefined procedures will have only short-term effects.

- Rasmussen suggests
  - The ability of the operator to explore should be supported by the system design.
  - A mean for recovery from the effects of errors should be provided.

  - Design for error tolerance [276, 274, 271]
    - Designing work conditions in which errors are immediately observable and reversible

  - Some ways to accomplish this goal, along with other approaches to enhancing safety through human-machine interface design, are described in Chapter 17.

5.4 Conclusions

- With automation increasingly taking over routine tasks, operators will use
  - Less skill-based and rule-based behavior
  - More knowledge-based behavior

- Traditional industries are highly reluctant to train operators beyond a limited list of specified responses to anticipated problems.
  - But, many incidents are examples of what can happen in complex systems when operator do not understand how equipment work and what they are required to do in an emergency.

- Human error is generally used in two different ways
  (1) The human did something that he should have known was wrong
  (2) The human did something that could not have been known at the time to be wrong but in retrospective was.

- Many authors suggested that the whole concept of human error as a cause of accidents is outdated and should be not used any more.

- The new question may be
  - How to prevent human-machine mismatches
  - How to design an interaction between human and machine

  - Allowing a symbiotic relationship enhancing the natural abilities and advantages of both.

Chapter 6

THE ROLE OF HUMANS IN AUTOMATED SYSTEMS

Introduction

The question today is not whether a function can be automated, but whether it should be, due to various human factors issues. It is highly questionable whether total system safety is always enhanced by allocating functions to automatic devices rather than human operators, and there is some reason to believe that flight-deck automation may have already passed its optimum point.

- Earl Weiner and Renwick Curry
Flight-Deck Automation: Promises and Problems

- Increased interest in human factors and the human-machine interface.
  - As human tasks are more and more being taken over by machines.

- Automation usually does not eliminate humans, but instead raises their tasks to new levels of complexity.
  - The easy parts may be eliminated, but leaving the difficult parts or increasing the difficulty of the remaining tasks.

- Volvo [146]
  - Our philosophy is that every operator or maintenance mistake that can be made will be made sooner or later. Therefore, we have taken all safety measures against human error. This approach is difficult to develop. It is much more convenient to stick to the idea of negligence.

- To take this approach, designers need to have a good knowledge of human behavior.

- It is now widely recognized that the TMI accident reflected a lack of appropriate design of the interaction between humans and machines.
  - If we are to avoid the trap of oversimplifying human error and do not attempt simply to replace humans in systems, then we need to determine their proper role in automated systems.
    - Human as Monitor
    - Human as Backup
    - Human as Partner

6.1 Mental Models

- In dealing with complexity, abstraction is one of our most powerful tools.
  - Abstraction involves forming mental models that provide predictive and explanatory power.
  - Mental models reflect individual goals and experience.

- The designer's model of a system may differ from the operator's model.
  - The designer's vision of the plant is often based on engineering or mathematical models of a control loop and is appropriate for situations where important decisions need not be made quickly.
  - Operators' model may be more mechanistic, since their goal is to keep the plant in a particular state. Their models need to be useful for making diagnoses of situations and for determining remedial action easily and quickly.

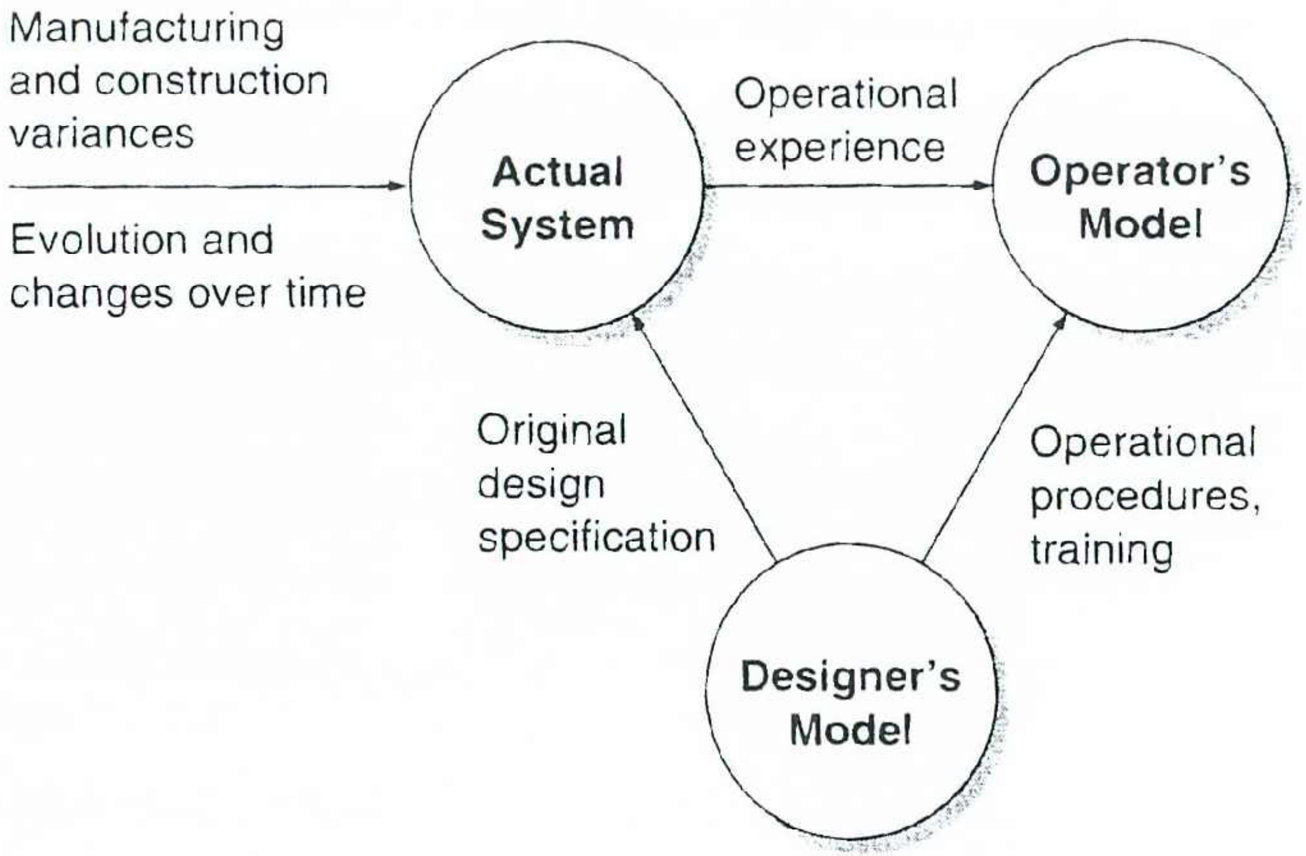

Manufacturing and construction variances
Evolution and changes over time
Actual System
Operational experience
Operator's Model
Original design specification
Operational procedures, training
Designer's Model

FIGURE 6.1
The relationship between mental models.

- Physical environments and system change.
  - However, operators don't change their models in the face of continued conflicts between the evidence and their perception of that is going on.

- This ability to change mental models is what makes the human operator so valuable.
  - Based on current inputs, the operators' actual behavior may differ from the prescribed procedures.
  - When the deviation is correct, then the operators are considered to be doing their job.
    - The designers' models are less accurate than the operators' models at that particular instant in time.
  - When the operators' models are incorrect, they are often blamed for unfortunate results.
    - Even though their incorrect mental models may have been reasonable given the information they had at the time.

6.2 The Human as Monitor

- With the automation of control systems, the operator’s role changes from active control to monitoring.
  - The human becomes responsible for detecting problems and providing a repair capability.
- A reasonable argument can be made that this change is an improvement.
  - Unfortunately, experience shows that humans make very poor monitors of automatic systems.

- There are several posited explanations for their poor performance.
  - The task may be impossible.
    - There is usually no way for a human to check in real time whether a computer is operating correctly or not.
  - The operator is dependent on the information provided.
    - It is easy to provide too much or too little information or to display it poorly.
  - The information is more indirect with automated systems, which may make it harder for the operator to obtain a clear picture of the state of the system.
  - Failures may be silent or masked.
  - Tasks that require little active operator behavior may result in lowered alertness and vigilance and can lead to complacency and overreliance on automated systems.

6.3 The Human as Backup

- Backup in the event of emergency.

- Poorly designed automation may make this role more difficult.
  - A poorly designed human-machine interface may leave operators with lowered proficiency and increased reluctance to intervene.
  - Fault-intolerant systems may lead to even larger errors.
    - Sometimes designers are so sure that their systems are self-regulating that they do not include appropriate means for human intervention.
  - The design of the automated system may make the system harder to manage during a crisis.

- In summary, automation can change the role of a human operator from an active controller to a passive monitor and occasional backup controller during emergencies
    - In addition, computers allows system to be built that are naturally unstable and more difficult to control, requiring more complex and faster control maneuvers.
    - At the same time
        - The opportunities for learning and practicing skills are decreasing,
        - The demands for those skills (on the rare occasion when they are needed) may be growing because of the steadily increasing complexity of the processes being controlled.

6.4 The Human as Partner

- The human and the automated system may both be assigned control tasks.
  - Unless this partnership is carefully planned, the operator may simply end up with the tasks that the designer cannot figure out how to automate.
    - The number of tasks that the operator must perform is reduced,
    - But, the error potential may be increased.

- Some issues are exemplified by the automation of air traffic control, and the Rand report cites two principal drawbacks.
  1. The design would not really allow a human backup in case the computer failed to handle a given situation correctly.
  2. Even if controllers had the time to handle conflicts, their passive role in the system would over time tend to make them unreliable monitors.

- How that partnership should be defined is an open question that highlights important technical problems.
  - In addition, a partnership of human and computer requires some means of communication between them.
  - Designing a partnership ventures into the trans-scientific domain.

- The human-machine interface become especially critical in an emergency, when the operator is under stress or is tired.
  - Much more attention to the design of the human-machine interface is needed, if we are to reduce accidents in automated systems.

6.5 Conclusions

- If the alternatives are considered, the unique abilities of human operators seem to overweight any disadvantages of including them in complex systems.
  - The goal of HMI design should be to preserve the human capability to intervene positively while making harmful intervention as difficult as possible.
  - The risk with respect to human operators may not be that they will cause accidents but that they may not succeed in preventing them.
- Unfortunately, in many cases, the operator is not considered a part of the system throughout the entire design process.
  - Instead, tackled on when the design is nearly completed.
- One way to solve this problem is to involve operators in safety analysis and design decisions throughout development.

Part Two

INTRODUCTION TO SYSTEM SAFETY

It is an easy task to formulate a plan of accident-preventing devices after the harm is done, but the wiser engineer foresees the possible weakness, as well as the dangers ahead, which are involved in his new enterprise, and at once embodies all the necessary means of safety in his original design.

- John H. Cooper, 1891

Chapter 7

FOUNDATIONS OF SYSTEM SAFETY

Introduction

> *Underlying every technology is at least one basic science, although the technology may be well developed long before the science emerges. Overlying every technical or civil system is a social system that provides purposes, goals, and decision criteria.*
>
> *- Ralph F. Miles, Jr.*
> *Systems Concepts: Lectures on Contemporary Approaches to Systems*

- System safety has its root in industrial safety engineering.
  - System engineering and system analysis after World War II

7.1 Safety Engineering Before World War II

- Humans have always been concerned about their safety.
  - Prior to the industrial age, natural disasters provided the biggest challenges.
  - Things started to change during the early part of the Industrial Revolution.

- Workers in the factories were considered expendable.
  - The prevailing attitude was that when people accepted employment, they accepted the risks involved in the job and should be smart enough to avoid danger.

- The horrible working conditions existing at this time led to social revolt by activists and union leaders.
  - Workers' compensation and security insurance
  - The main concern of labor unions was the safety and health of their member, and they demanded federal occupational safety and health legislation.

- In 1908, the state of New York passed the first workers’ compensation law.
  - Required management to pay for injuries that occurred on the job regardless of fault.

- When management found that it had to pay for injuries on the job,
  - Effort was put into preventing such injuries.
  - The organized industrial safety movement was born.

- A few engineers had recognized the need to prevent hazards early in the industrial era,
  - When the machines they were designing and building began to kill people.

  - James Watt had warned about the dangers of high-pressure steam engines in the early 1800s.

- Near the end of the 19th century, engineers began to consider safety, as well as functionality, in their designs, instead of simply trying to add it on in the form of guards.
  - The first organization: the Prevention of Accidents in Factories
  - The engineering technical literature started to acknowledge that safety should be built into a design [293].
  - In 1899, "The Prevention of Factory Accidents" was published in England.
  - By 1911, there was a calling for legislation to force manufacturers to provide safe product.
  - In 1914, the first safety standards published in the US, the Universal Safety Standards. According to them, a safeguard should
    - Afford all possible safety to the operator and surrounding workmen
    - Be automatic in its action, application, or operation (if possible)
    - Be an integral part of the machine itself (if possible)
    - Not materially diminish the output or efficiency of the machine to which it is applied
  - In 1929, engineers and others started to study safety as an independent topic.

- In the early years of the safety movement, emphasis was concentrated on hazardous physical conditions.
  - Correcting these conditions produced remarkable results during the first 20 years.
  - Heinrich was the first to provide a conceptual model upon which to build a theoretical framework for industrial safety.

- World War II led to even greater safety efforts than had previously been attempted.
  - Along with the vastly increased war manufacturing came a tremendous increase in accident and injury rates.

- The post-war period also gave rise to a new approach to safety based on *system engineering principles*.
  - Most safety programs had been established on an *a posteriori* philosophy: After an accident, an investigation was conducted to determine how to prevent a similar accident in the future.
  - The increased complexity and cost of new products made it uneconomical to follow the accident-investigate-fix philosophy [294].

- The rapid expansion of technology following World War II caused American industry to begin to think in terms of *systems*.

7.2 Systems Theory

- *System theory* is a complementary approach and reaction to scientific reductionism.
  - While reduction, repeatability, and refutation together form the basis of the scientific method.
  - In system theory, problems are divided into distinct parts and the parts are examined separately.

- The principle of reductionism makes three important assumptions [49].
  1. The division into parts will not distort the phenomenon being studied.
  2. The components of the whole are the same when examined singly as when they are playing their part in the whole.
  3. The principles governing the assembling of the components into the whole are straightforward.

- Three types of systems

  (1) Organized Simplicity

  (2) Unorganized Complexity

  (3) Organized Complexity

- Organized simplicity
  - The precise nature of the interactions is known.
  - Component interactions can be examined pairwise.
  - System can be separated into non-interacting subsystems for analysis purpose without distorting the results.
  - In physics, this approach has been highly effective and is embodied in structural mechanics.

- Unorganized complexity
  - These systems lack the underlying structure that allows reductionism to be effective.
  - They are complex but regular and random enough in their behavior that they can be studied statistically.
  - The basis of this approach is the law of large numbers.
    - The larger the population, the more likely that observed values are close to the predicted average values.

- Organized complexity
  - These systems are too complex for complete analysis and too organized for statistics.
  - The averages are deranged by the underlying structure.
  - It describes many of the complex engineered systems of the post-World War II era.
  - It also represents particularly well the problems that are faced by those attempting to build complex software.

- The systems approach focuses on systems taken as a whole, not on their parts taken separately.
  - The systems approach concentrates on the analysis and design of the whole as distinct from the components or the parts.

- *Systems theory* starts from some basic definitions.
  - A *system* is a set of components that act together as a whole to achieve some common goal, objective, or end.
  - *Components* are all interrelated and either directly or indirectly connected to each other.

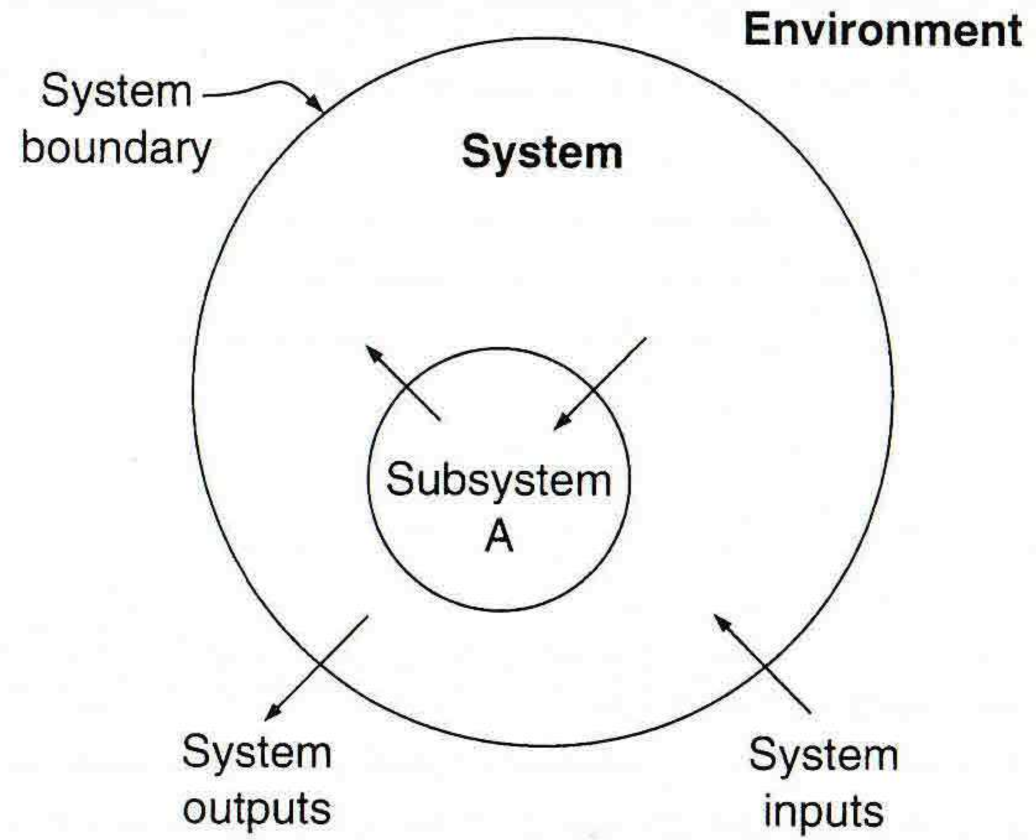


FIGURE 7.1
Definition of a system.

- System state
  - The set of relevant properties describing the system at that time
- System environment
  - A set of components (and their properties) that are not part of the system but whose behavior can affect the system state
- Inputs and Outputs
  - Anything that crosses the boundary between the system and its environment implicitly
- Subsystem
  - A part of a larger system

- A system is always a model – an abstraction conceived by the analyst.
  - For the same system, an observer may see a different purpose that the designer had, and may focus on different relevant properties.

- Systems theory provides a means for studying systems exhibiting organized complexity.

- The foundation of systems theory is composed of two pairs of ideas
  - (1) Emergence and Hierarchy
  - (2) Communication and Control

7.2.1 Emergency and Hierarchy

- A general model of organized complexity can be expressed in terms of a *hierarchy* of levels,
  - Each more complex than the one below,
  - Where a level is characterized by having *emergent* properties.

- The concept of emergence is the idea that
  - At a given level of complexity, some properties characteristics of that level are irreducible.
  - Emergent properties associated with a set of components at one level in a hierarchy are related to constraints upon the degree of freedom of those components.

- Safety is an emergent property of systems.
  - Safety can only be determined by the relationship between one component and others - that is, in the context of the whole.

7.2.2 Communication and Control

- The second major pair of ideas in systems theory is *communication* and *control*.
  - The implication of constrains upon the activity at one level of hierarchy is an example of regulatory or control action.
  - Hierarchies are characterized by control processes operating at the interfaces between levels.

- Control in open systems implies the need for communication.
  - Open systems have inputs and outputs from their environment.
    - Closed systems: Unchanging components settle into a state of equilibrium
    - Open systems: Can be thrown out of equilibrium by exchanges with their environment
  - Open systems can achieve a steady state that depends upon continuous exchanges with an environment.
  - It also applies to man-made systems, such as a factory.

- The plant controller
  - Obtain information about the process state from measured variables,
  - Uses this information to initiate action by manipulating control variables to keep the process operating within predefined limits despite disturbances to the process.

- In general, the maintenance of any open-system hierarchy will require a set of processes in which there is communication of information for regulation or control [49].

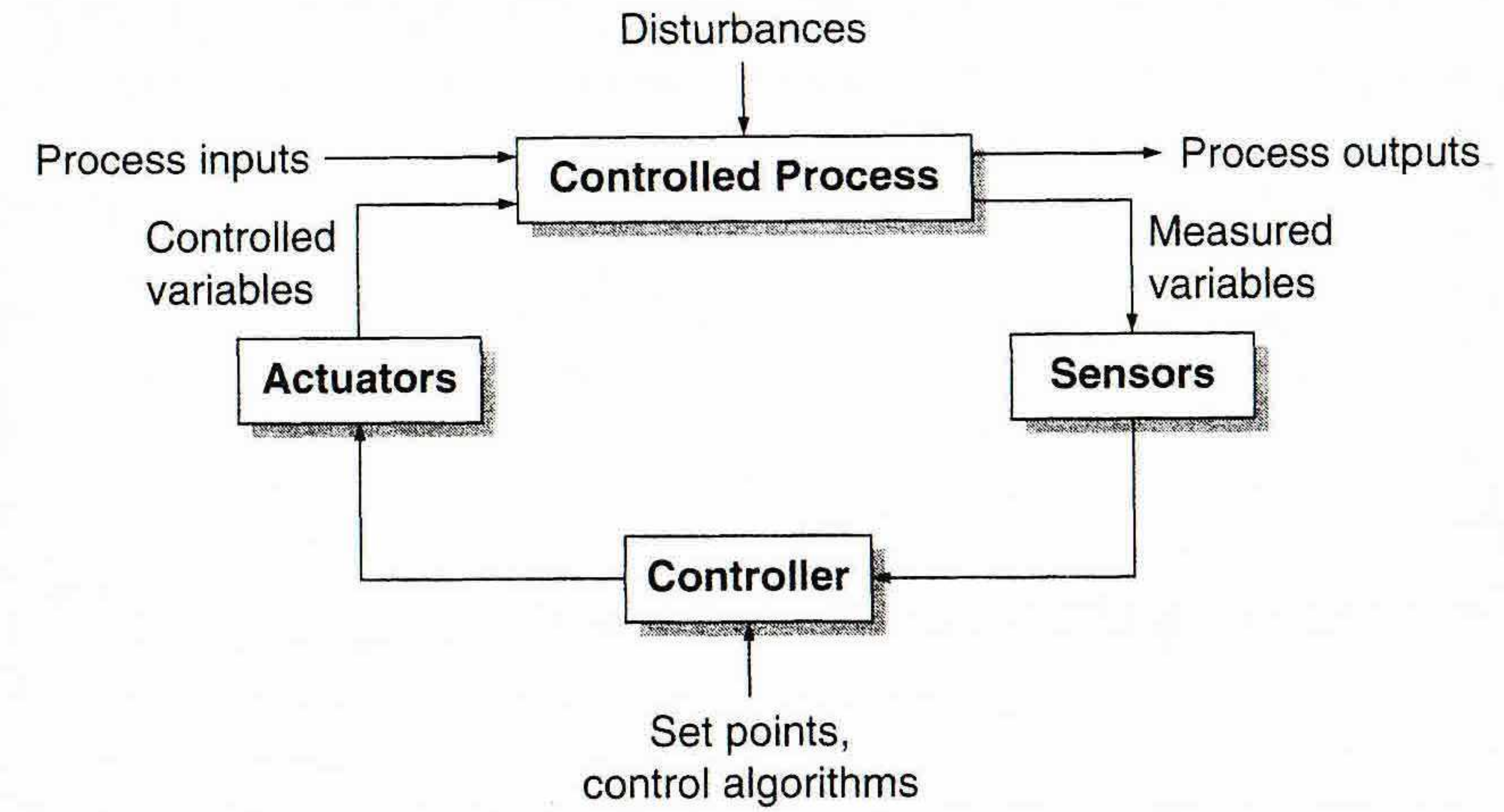


FIGURE 7.2
A standard control loop.

7.3 Systems Engineering

- A new emphasis in engineering, called '*systems engineering*'

- During and after World War II, technology expanded rapidly and engineers were faced with designing and building more complex systems than had been attempted previously.
  - *Large* in number of parts, replication of identical parts, functions performed, and cost
  - *Complex*, in terms of a change in one variable affecting many other variables, usually in a nonlinear fashion
  - *Semi-automatic* with a human-machine interface, where machines perform some functions while humans perform others
  - *Unpredictable* (random) in terms of the inputs, the rate of inputs, or other environmental disturbances

- System theory provides a theoretical foundation and approach for systems engineering,
  - Concerned with optimizing the design and development of an overall system.

- The development of systems engineering as a discipline enabled the solution of enormously more complex and difficult technological problems than before [221].

- Systems engineering is more a shift in emphasis than a change in content. More emphasis is placed on
  1. Defining goals and relating system performance to these goals
  2. Establishing and using decision criteria
  3. Developing alternatives
  4. Modeling systems for analysis
  5. Controlling and managing implementation and operation

- While much of engineering is based on technology and science,
  - Systems engineering is equally concerned with overall management of the engineering process.

- A goal of systems engineering is to optimize the system operation according to prioritized design criteria.
  - System is composed of diverse, specialized components.
  - System may have multiple objectives and some of these may partially conflict.

- The basic assumption underlying systems engineering
  - Optimization of individual components or subsystems will not lead in general to a system optimum.
  - In fact, improvement of a particular subsystem actually may worsen the overall system.
  - Since every system is merely a subsystem of some larger system, it's a difficult problem.

- The systems approach provides a logical structure for problem solving.

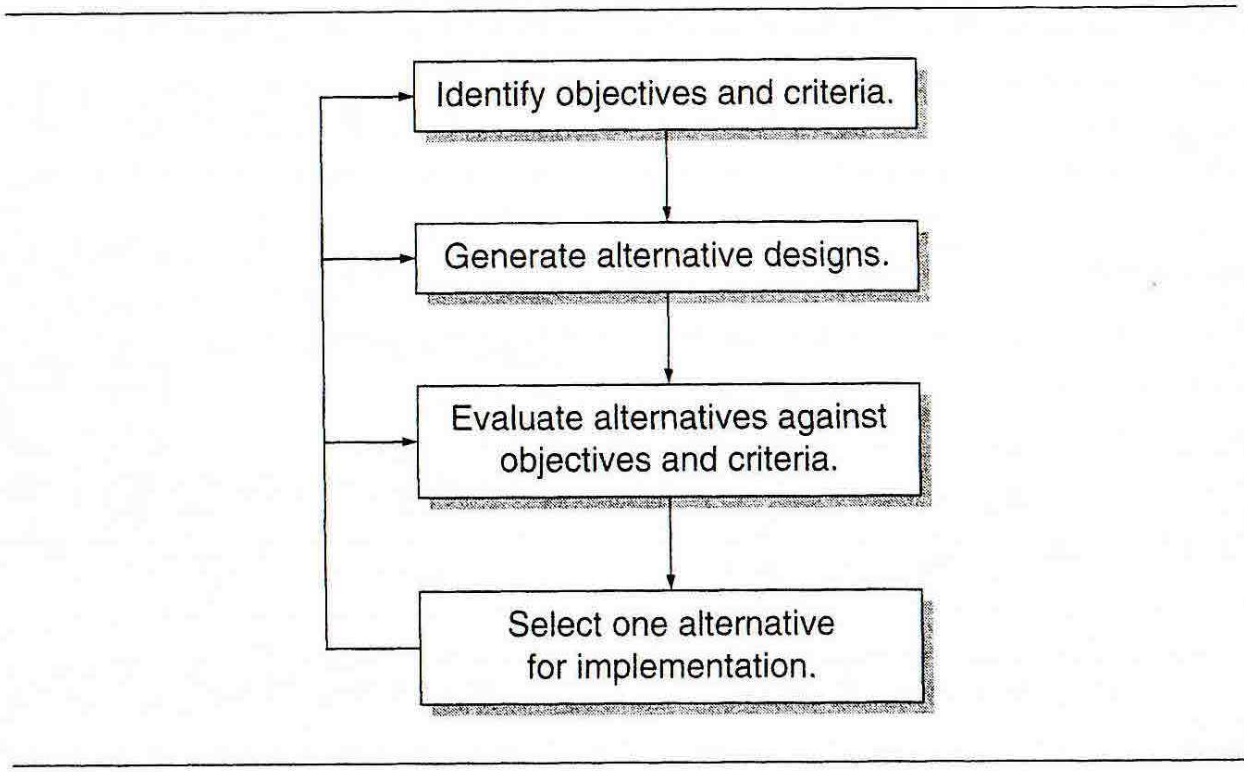

Identify objectives and criteria.
Generate alternative designs.
Evaluate alternatives against objectives and criteria.
Select one alternative for implementation.

FIGURE 7.3
The problem-solving model for systems engineering.

- System engineers need not be experts in all the aspects of the system.
  - System engineering often requires a team effort.
  - Some of the basic components of this process are
    - Need analysis
    - Feasibility studies
    - Trade studies
    - System architecture development and analysis
    - Interface analysis

7.4 Systems Analysis

- During the 1950s, the RAND Corporation developed a methodology called '*system analysis*' to provide a rational way to evaluate the alternatives facing a decision maker.
  - System analysis is a method for broad economic appraisal of the costs and consequences of the various ways to satisfy a particular goal.
- In most cases, all elements of uncertainty cannot be eliminated from the decision process.
  - Because all the relevant information will not be obtainable.
  - Consequence of a particular course of actions cannot be determined completely.
  - However, systems analysis provides an organized process for the acquisition and investigation of specific information pertinent to a given decision.
- Systems engineering and systems analysis have merged over the years.
  - These two disciplines provide the foundation for system safety.

Chapter 8

FUNDAMENTALS OF SYSTEM SAFETY

Introduction

My company has had a safety program for 150 years. The program was instituted as a result of a French law requiring an explosives manufacturer to live on the premises with his family.

- Crawford Greenwalt
Former president of Dupont

- After World War II,
  - System safety developed in parallel with and in conjunction with systems engineering.
  - Many of basic concepts and approaches of system safety became a recognized discipline.

8.1 Historical Development

- Much of the early development of system safety as a separate discipline began with flight engineers immediately after World War II.
  - Many examples

- Building many of today's complex systems requires integrating parts built by separate contractors or organizational entities.

- It became evident in many industries that designing safety into a plant or product could reduce overall life-cycle costs.
  - A system safety approach was required to achieve acceptable levels of safety.

8.2 Basic Concepts

- System safety uses *systems theory* and *systems engineering* approaches to
  - Prevent foreseeable accidents and minimize the result of unforeseen ones.
- The primary concern of system safety is the management of hazards.
  - Their identification, evaluation, elimination and control through analysis, design and management procedures
- System safety activities start in the earliest concept development stages of a project and continue through design, production, testing, operational use and disposal.

- Some general principles distinguish system safety from other approaches to safety and risk management.
  - System safety emphasizes building in safety, not adding it on to a completed design.
  - System safety deals with systems as a whole rather than with subsystems or components.
  - System safety takes a longer view of hazards than just failures.
  - System safety emphasizes analysis rather than past experience and standards.
  - System safety emphasizes qualitative rather than quantitative approaches.
  - System safety recognizes the importance of tradeoffs and conflicts in system design.
  - System safety is more than just system engineering.

- System safety attempts to manage hazards through analysis, design, and management procedures, using these general principles.

8.2.1 Hazard Analysis

- Hazard analysis investigates factors related to accidents.

- Hazard analyses have been used in
  (1) Development to identify and assess potential hazards, and the conditions that can lead to them, so that the hazards can be eliminated or controlled
  (2) Operations to examine an existing system to improve its safety and to formulate policy and operational procedures
  (3) Licensing to examine a planned or existing system to demonstrate acceptable safety to a regulatory authority

- The type of analysis will vary with the purpose of the analysis
  - Techniques to engineer safer systems:
    - Used to make something safer (Emphasis in this book)
  - Risk assessment
    - Used to convince a government licenser that it already is safe

- Four categories of system safety analysis (hazard analysis)
  - (1) Preliminary Hazard Analysis (PHA)
  - (2) System Hazard Analysis (SHA)
  - (3) Subsystem Hazard Analysis (SSHA)
  - (4) Operating and Support Hazard Analysis (OSHA)

- Preliminary Hazard Analysis (PHA)
  - Used in the early life cycle stages to identify critical system functions and broad system hazards.
    - Identified hazards are often assessed and prioritized.
    - Safety design criteria and requirements may be identified.
  - Started early in the concept exploration phase or in the earliest life-cycle phases of the program.
  - Safety considerations are included in tradeoff studies and design alternatives.
  - Iterative process
  - Little detail is available.
    - Because PHA starts at the concept formation stage of a project.
    - Assessments of hazard and risk levels are necessarily qualitative and limited.

- System Hazard Analysis (SHA)
  - To recommend changes and controls, and evaluate design responses to safety requirements.
  - Begins as the design matures (around preliminary design review) and continues as the design is updated and changes are made.
  - Involves detailed studies of possible hazards created in the interfaces between subsystems or by the system operating as a whole, including human errors.
  - Examines subsystem interfaces for
    - Compliance with safety criteria in the system requirements specification
    - Degradation of safety resulting from normal operation of the system and subsystem
    - Possible combinations of independent, dependent, and simultaneous hazardous events or failures that could cause hazards

- Subsystem Hazard Analysis (SSHA)
  - Started as soon as the subsystems are designed in sufficient detail.
  - To identify hazards associated with the design of the subsystems, including
    - Operating modes (normal or failure modes)
    - Critical inputs and outputs
    - Hazards resulting from functional relationships between the components and equipment comprising each subsystem
    - Normal performance and performance degradation
    - Functional failure, inadvertent functioning, etc.
  - Software Hazard Analysis is a type of SSHA.
  - SHA and SSHA are different in the goals.
    - SSHA examines how individual component operation or failure affects the overall safety of the system.
    - SHA determines how normal and failure modes of the components operating together can affect system safety.

- Operating and Support Hazard Analysis (OSHA)
  - To identify hazards and risk reduction procedures during all phases of system use and maintenance
  - Especially examines hazards created by the human-machine interface.

- Hazard analysis is discussed in depth in later chapters.

8.2.2 Design for Safety

- Once hazards have been *identified* by hazard analyses, the highest priority should be assigned to *eliminate* them.
  1. Hazard elimination (intrinsic safety)
  2. Hazard reduction
     - By designing in hazard reduction measures, such as lockouts, lockins, or interlocks.
  3. Hazard control
     - Using devices to control a hazard, once it has occurred.
     - Controls may be passive and active.
       - Passive controls rely on basic physical principles, e.g., gravity.
       - Active controls require some action to prevent or mitigate the hazard's effects.
  4. Damage reduction
     - In case the hazard condition measures are ineffective or unidentified hazards occur
       - Warning devices, training, procedures, and isolation of the hazardous system from population centers

- The higher levels are more desirable.
  - A design very likely will include features at several levels.

8.2.3 Management

- The most important aspect of system safety in terms of accident prevention may be management procedures.
  - Setting policy and defining goals
  - Planning tasks and procedures
  - Defining responsibility
  - Granting authority
  - Fixing accountability
  - Documenting and tracking hazards and their resolution
  - Maintaining safety information systems and documentation
  - Establishing reporting and information channels and so on.

- System safety is responsible for system-level safety effort, including analysis of component interfaces.
  - Component-level safety activities may be part of the general system safety responsibility.
  - Even if safety efforts are subdivided, system safety engineers are still responsible for integrating the component safety activities and information.

- System safety also has obvious interface with related disciplines.
  - Reliability engineering
  - Quality assurance
  - Human factor

8.3 Software System Safety

- Increased use of computers in safety-critical applications presents new problems.
  - – Methods for handling the software aspect of system safety have not yet been developed to the extent necessary to ensure safety in these systems.
  - – A chasm or disconnect between software engineering and system or safety engineering
    - System engineers ignore software or treat the computer as a black box.
    - Software engineers treat the computer as a stimulus-response system.

- The computer serves a control function in many systems.
  - – Software plays a direct and important role in system safety and must be an integral part of the system safety efforts.

Definition. *Software system safety* implies that the software will execute within a system context without contributing hazards.

- Two software system safety issues must be handled.
  (1) Software can exhibit behavior in terms of output values and timing that contribute to the system reaching a hazardous state.
  (2) Software can fail to recognize or handle hardware failures that it is required to control or to respond to in some way.

Definition. *Safety-critical software* is any software that can directly or indirectly contribute to the occurrence of a hazardous system state.

Definition. *Safety-critical functions* are those system functions whose correct operation, incorrect operation (including correct operation at the wrong time), or lack of operation could contribute to a system hazard.

Definition. *Safety-critical software functions* are those software functions that can directly or indirectly, in consort with other system component behavior or environmental conditions, contribute to the existence of a hazardous state.

- Software does not exhibit random wear-out failures as does hardware.
  - It is an abstraction.
  - Failures are due to logic or design errors.

- Software is merely the design of a machine.
  - It is an abstraction like an electrical engineer's circuit diagrams.
  - Has no physical reality
  - When the software design is executed on a general-purpose computer, the special-purpose machine (as reflected in the software) becomes a concrete machine.

- Therefore, software can and must be evaluated both as an abstract design and as a concrete machine.

- Firmware is just software.
  - A layer of abstraction which has been inserted between the digital hardware and the application software
  - Simply a software (instructions) that controls the operation of a general-purpose machine

- A computer can behave incorrectly because of logic errors in the software - in the design of the special-purpose machine.

  (1) Logic error
  - The software is written from incorrect requirements.
  - The code matches the requirements, but the behavior specified in the requirements is not desired from a system perspective.

  (2) Programming error
  - The implementation of the requirements in a programming language does not match or satisfy the requirements.

- Both types of errors must be considered to increase software reliability and safety.
  - The majority of the existing software engineering techniques deal only with the second problem.

- However, the majority of safety problems arise from software requirements errors and not coding errors.

- Three ways to deal with software errors
  (1) Just to get the requirements and code 'correct'
    - Theoretically possible, it is impossible from a practical standpoint.
    - It requires building software to do exactly what it should under all conditions.

  (2) Attempting to make the software fault tolerant thorough various types of redundancy
    - It would provide protection against coding errors.
    - Reliability and safety are sometimes at odds with respect to critical functions.

  (3) Applying standard system safety techniques

- These first two approaches attempt to increase safety by increasing operational software reliability.
  - Merely increasing reliability may not be adequate to ensure safety.
  - Software related accidents have occurred when the software satisfied its specification and when the operational reliability of the software was very high.
    - The software correctly implements the requirements, but the requirements specify behavior that is not safe from a system perspective.
    - The requirements do not specify some particular behavior that is required for system safety.
    - The software has unintended and unsafe behavior beyond what is specified in the requirements.

- The particular software behaviors that can lead to system hazards should be *identified* and *eliminated* or *controlled* in the software and system requirements and code.
  - ***Safeware*** approach to software system safety

8.4 Cost and Effectiveness of System Safety

- To be practical, system safety approaches must be effective and have a comparatively reasonable cost.
  - Obviously, the investment in an aviation safety program is worthwhile if it prevents the loss of a single aircraft ($25 million for the F-18 and over $2 billion for the B-2), not to mention the loss of human lives.
  - The same ratio applies to other industries.

- The payoff occurs, only if accidents are prevented.
  - The evidence for effectiveness of system safety programs is harder to obtain.
  - Measuring something that does not happen is difficult.

- Three ways of assessing the effectiveness of system safety programs
  (1) Comparing systems with and without safety programs [88]
  (2) Looking at many hazards that system safety personnel corrected before accidents occurred.
  (3) Examining cases where system safety recommendations were not followed and an accident occurred.

8.5 Other Approaches to Safety

- Industrial (or operational) safety and reliability engineering are related to, but differ from, system safety.
  - The relationships and differences are important in understanding system safety and the approach to increasing safety.

8.5.1 Industrial Safety

- Industrial (or occupational) safety has been called the “hard hat and safety shoe” philosophy.
  - The industrial safety engineer usually is dealing with a fixed manufacturing design and hazards that have existed for a long time.
  - Many of which are accepted as necessary for operations.
  - More emphasis is placed on teaching employees to work within this environment than on removing the hazards.

- Industrial safety activities are designed to protect workers in an industrial environment.
  - Extensive standards are imposed by federal codes or regulations providing for a safe workplace.

- System safety attempts to identify potential hazards before the system is designed,
  - To define and incorporate safety design criteria and
  - To build safety into the design before the system becomes operational.
- More emphasis is placed on upfront analysis and designing for safety.

8.5.2 Reliability Engineering

- Reliability engineering is concerned primarily with failure and failure rate reduction [108].
  - The reliability engineering approach to safety concentrates on failures as the cause of accidents.

Definition. *Reliability* is the characteristic of an item expressed by the probability that it will perform its required function in the specified manner over a given time period and under specified or assumed conditions.

- Reliability engineering uses a variety of techniques to minimize component failures
  - Parallel redundancy
  - Standby sparing
  - Safety factors and margins
  - Derating
  - Screening
  - Timed replacements

- Parallel redundancy
  - The same functions are performed by two or more components at the same time.
  - Often with some type of voting mechanism to determine which output to use
  - If one unit fails, the function can still be performed by the remaining units.

- Standby sparing
  - Inoperative or idling standby units take over, if an operating unit fails.

- Safety factors and margins
  - An item is designed to be several times stronger than is necessary to withstand the expected stress.

- Derating
  - The maximum operational stress ratings of components are reduced.
  - Failures occur in electronic components under specific conditions and stress.

- Screening
  - Components are eliminated that pass operating tests for certain parameters, but with indications that they will fail within an unacceptable time.

- Time replacements
  - Components are replaced before they wear out.

- While these techniques are often effective in increasing reliability, they do not necessarily increase safety.
  - In fact, their use under some conditions may actually reduce safety.

- Safety and reliability are overlapping, but are not identical.

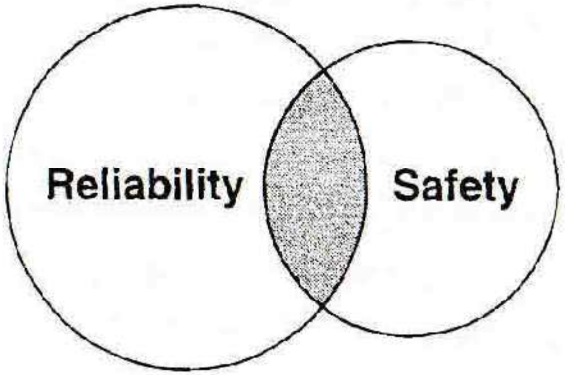


FIGURE 8.1
Safety and reliability are overlapping, but are not identical.

- Safety is an emergent property.
  - Arising at the system level, when components are operating together.
  - The events leading to an accident may be a complex combination of
    - Equipment failure
    - Faulty maintenance
    - Instrumentation and control problems
    - Human actions and
    - Design errors.

- Reliability analysis considers only the possibility of accidents related to failures.
  - Not investigating potential damage that could result from successful operation of the individual components.

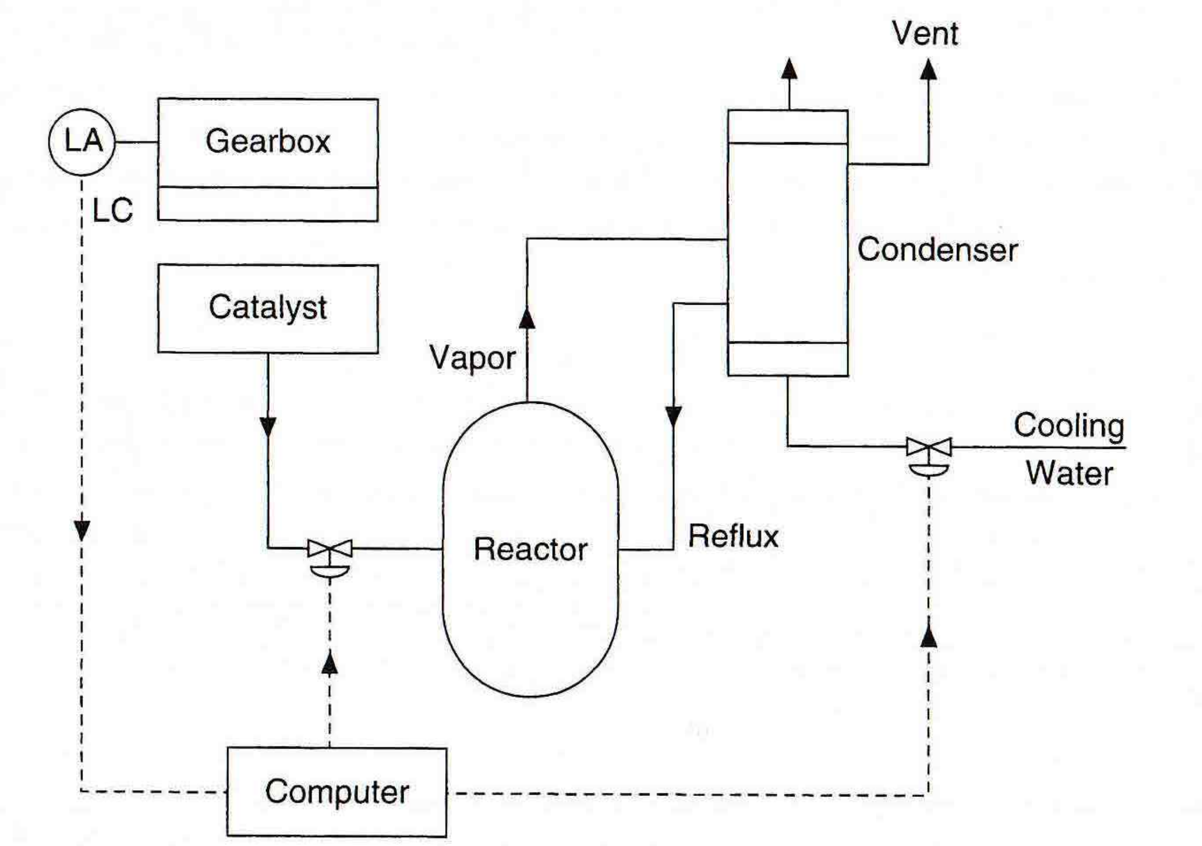


FIGURE 8.2
A chemical reactor design. (Source: Trevor Kletz, Human problems with computer control, *Plant/Operations Progress*, 1(4), October 1982. Reproduced by permission of the American Institute of Chemical Engineers. ©1982 AIChE. All rights reserved.)

- Reliability engineering has an important role to play in reducing random failures in hardware.

- But, they will not necessarily increase safety.
  - Sometimes measures taken to increase individual component reliability will reduce safety.
  - Reliability and safety may conflict.
  - Many examples in the text.

- While reliability engineering cannot replace system safety, it can supplement it.

- Care needs to be taken when applying reliability assessment techniques to safety.
  - Since, accidents are not necessarily caused by events that can be measured this way, it should not be used as a measure of risk.

- The major drawbacks in reliability models are often not what they include but what they do not include.

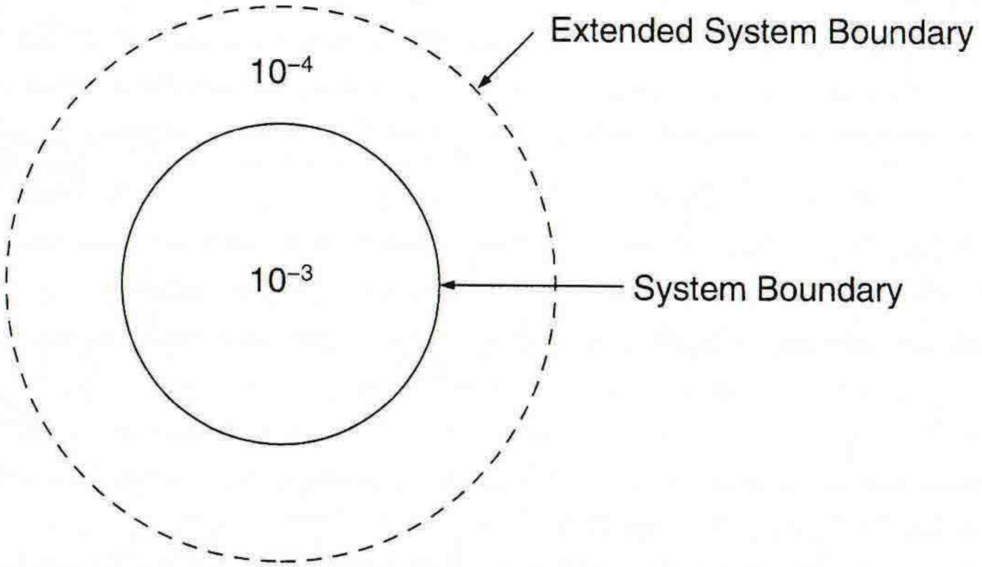


FIGURE 8.3
The bounding and resolution problem in system modeling. (Source: W. E. Veseley, F. F. Goldberg, N. H. Roberts, and D. F. Haasl. *Fault Tree Handbook*, Technical Report NUREG-0492, U.S. Nuclear Regulatory Commission, Washington, D.C., 1981, pages I–7.)

- Reliability assessments based on hardware failure rates provide information about which random events are most likely and therefore where failure reduction resources should be applied to reduce hazardous random hardware failures.

- But, assessing failure rates should not be confused with assessing hazards or risk of accidents.

- "Distinguishing hazards from failures is implicit in understanding the difference between safety and reliability." [223]

Part Three

DEFINITIONS AND MODELS

> *"When I use a word," Humpty Dumpty said, in rather a scornful tone, "it means just exactly what I choose it to mean – neither more nor less."*
>
> *-Lewis Carroll*
> *Through the Looking Grass*

Chapter 9

TERMINOLOGY

Introduction

> *Defining concepts is frequently treated by scientists as an annoying necessity to be completed as quickly and thoughtlessly as possible. A consequence of this disinclination to define is often research carried out like surgery performed with dull instruments. The surgeon has to work harder, the patient to suffer more, and the chances for success are decreased.*
>
> *- Russell L. Ackoff*
> *Towards a System of Systems Concepts*

- The goal of this chapter is to establish the definitions of a few basic terms that are used in this book.
  - Failure
  - Accident
  - Hazard
  - Risk
  - Safety

  - Also to differentiate safety from related qualities

- This book's definitions are consistent with engineering terminology.
  - Though they may conflict with computer science usage.

9.1 Failure and Error

Definition. *Reliability* is the probability that a piece of equipment or component will perform its intended function satisfactorily for a prescribed time and under stipulated environmental conditions.

- Unreliability is the probability of failures.

Definition. *Failure* is the nonperformance or inability of the system or component to perform its intended function for a specified time under specified environmental conditions.

- Two causes of failures in physical devices
  (1) Caused by design flaws → systemic failures
  (2) Resulted from a deviation from the originally designed behavior

- Both of these types will be categorized as failures in this text.

Definition. An *error* is a design flaw or deviation from a desired or intended state.

- A failure is defined as an event (a behavior).
  - Occurs when designs are realized in concrete devices which are operated.
- An error is defined as a static condition (a state).
  - An error or erroneous state may lead to an operational failure.
- Software itself does not fail.
  - It is a design for a machine, not a machine or a physical device.
  - However, the computer on which the software is executing may fail.
    - Computer hardware failures may be wear-out failures or systemic failures.
    - Software-related computer failures are always systemic.

- Fault vs. Failure
  - Engineers distinguish between a fault and a failure.
    - Failures are basic abnormal occurrence.
    - Faults are higher-order events.
  - "All failures are faults, but not all faults are failures."
  - Distinctions in faults
    - Primary fault
      - A component fails within the design envelope or environment.
      - Most often, it is caused by defective design, manufacture or construction.
    - Secondary fault
      - A component fails because of excessive environmental stress that exceed the requirements specification or design environment.
    - Command fault
      - Inadvertent operation of the component because of a failure of a control element.
      - The component operates correctly, but at the wrong time or place.

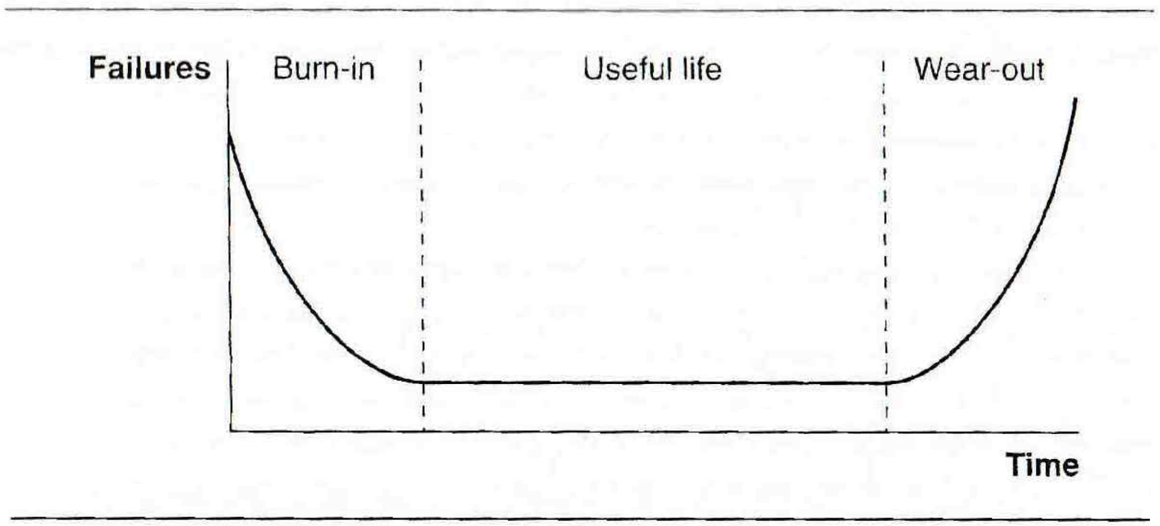


FIGURE 9.1
"Bathtub" model of reliability for electronic components, so-called because of its shape. Mechanical components tend not to exhibit the same type of constant failure rates over most of their lifetime and thus are more V-shaped.

- Failures
  - Early failures
  - Random or chance failures
  - Wearout failures

- Early failures
  - Occur during a debugging or burn-in period
  - Applied both to software and hardware

- Random or chance failures
  - Occur in the useful life of the component or system
  - An assumption: Random failures exist for software too.
    - Application of the assumption model to the software alone makes no sense.
    - Reliability measurement applies only when the software is executed on a particular computer.

- Wearout failures
  - Primarily due to hardware failure, not software
  - In case of software maintenance and modification
    - Thought to have a new design and under-go another burn-in period.

9.2 Accident and Incident

- The definition of an accident event is important because it influences the approach taken to increase safety.

Definition. An *accident* is an undesirable, an unplanned (but not necessarily unexpected) event that results in (at least) a specific level of loss.

- There are several important aspects of this definition.
  - Undesired
  - Specific level of loss
  - Loss event
    - Important not to limit the possible solution space of a problem by the definition itself.

Definition. A *near-miss* or *incident* is an event that involves no loss (or only minor loss), but with the potential for loss under different circumstances.

9.3 Hazard

Definition. A *hazard* is a state or set of conditions of a system (or an object) that, together with other conditions in the environment of the system (or object), will lead inevitably to an accident (loss event).

- A hazard is defined with respect to the environment of the system or component.
  - What constitutes a hazard depends upon where the boundaries of the system are drawn.

- The definition of hazard is arbitrary.
  - One of the first steps in designing a system is to decide what conditions will be considered to be hazards that need to be eliminated or controlled.

- A hazard has two important characteristics
  (1) Severity
  (2) Likelihood of occurrence
    - Almost few information is available to evaluate the hazard quantitatively.
    - Qualitative evaluation of likelihood is usually the best that can be done.

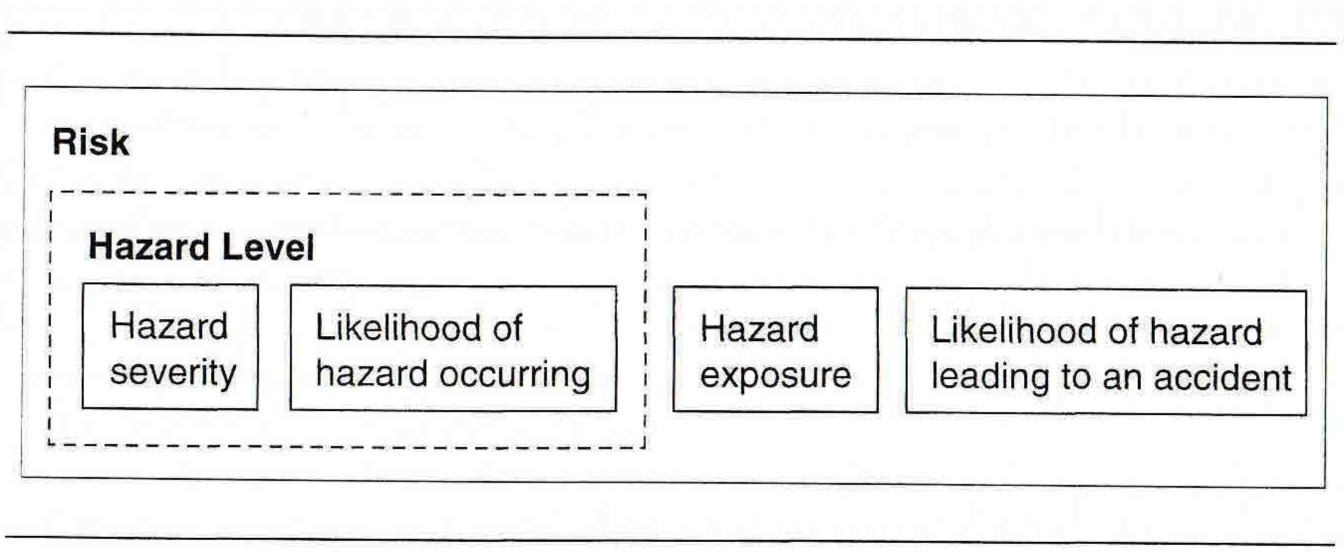

Risk
Hazard Level
Hazard severity
Likelihood of hazard occurring
Hazard exposure
Likelihood of hazard leading to an accident

FIGURE 9.2
The components of risk.

9.4 Risk

Definition. *Risk* is the *hazard level* combined with (1) the likelihood of the hazard leading to an accident (sometimes called *danger*) and (2) hazard exposure or duration (sometimes called *latency*).

- Exposure or duration of a hazard is a component of risk
  - An accident involves a coincidence of conditions.
  - The longer the hazardous state exists, the greater the chance that the other prerequisite conditions will occur. (if the hazard is just one)
- We can define a very simple risk model [61] defining risk as a function.
  - The correct way to combine the elements of the risk function is unknown.
  - Ad hoc quantitative way should involve qualitative judgment and personal values. (therefore trans-scientific)

- Hazard analysis is a subset of risk analysis
  - Hazard analysis involves only the identification of hazards and the assessment of hazard level.

- Risk analysis adds the identification and assessment of the environmental conditions along with exposure or duration.

- Reliability measurement is often used incorrectly as a measure of risk.
  - Reliability is a component property.
  - Risk or Safety cannot be defined or measured without considering the environment.

- Because component failure is the most convenient thing to measure, we often use it as measures of risk or assign it too much importance in risk assessment.
  - Most accidents in complex systems involve factors other than single component failure.

9.5 Safety

Definition. *Safety* is freedom from accidents or losses.

- We restrict the definition to identified hazards and conditions.
  - Therefore, it makes sense to talk about absolute safety from a particular hazard.

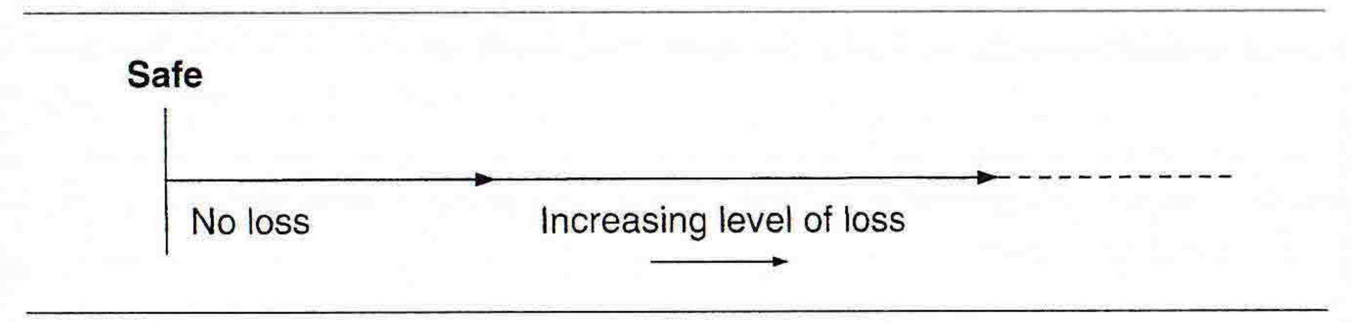


FIGURE 9.3
Safety as a continuum.

9.6 Safety and Security

- Safety is a distinct quality and should be treated separately from others.

- Safety and Security, however, are closely related.
  - Their similarities can be used to borrowing effective techniques from each.
  - Similarities
    - Both qualities deal with threats or risks.
    - Both often involve negative requirements or constraints that may conflict with some important system goals.
    - Both involve protection against losses.
    - Both involve global system properties that are difficult to deal with outside of a system context.
    - Both involve requirements that are considered of supreme importance.
    - Both involve aspects of a system that are regulated by government agencies or licensing bureaus.
  - Differences
    - Malicious actions vs. well-intended actions
    - Preventing unauthorized access to classified information vs. preventing more general malicious activities

- If an accident or loss event is defined to include unauthorized disclosure, modification, and withholding of data, then security becomes a subset of safety.

- Attempt to integrate several qualities into one abstraction seem misguided.
  - Dependability (combination of reliability, safety, security and availability)
  - Because, qualities conflict, and de facto tradeoffs are lost in global abstractions or measurements.

9.7 Summary

- Terminology is important for communication and progress in finding solutions to our problems.
  - Definitions can have powerful effects on the way we express our problems and how we go about solving them.

- Establishing a common terminology is always painful but is worth the effort in the long run.

Chapter 10

ACCIDENT AND HUMAN ERROR MODELS

Introduction

> *Accidents on the whole are becoming less and less attributable to a single cause, more to a number of contributory factors. This is the result of the skill of the designers in anticipating trouble, but it means that when trouble does occur, it is inevitably complicated.*
>
> *- DeHavilland and Walker*
> *(after reviewing failures of the Comet aircraft)*

- The design and analysis methods used for safety critical systems are based on particular underlying models of the accident process and human errors.
  - How effective our procedures are depending on how accurate our models are.
    - That is how well they reflect the features of the environment to which they are applied.

10.1 Accident Models

- Various accident models have been devised.
  - To understand past accidents and learn how to prevent future ones

- The power and features of the model used will greatly affect our ability to identify and control hazards and thus prevent accidents.

- There is a very large number of models.

- One way of understanding and comparing these models is to examine
  - Underlying accident mechanism
  - Types of causal factors
  - Relationship among the causal factors

10.1.1 Basic Energy Models

- The oldest and most pervasive model
  - Accidents as the result of an uncontrolled and undesired release of energy
  - Types of energy is an important variable in predicting and explaining.
  - If accidents are the result of uncontrolled energy flows, then the obvious way to reduce them is to use barriers or other flow control mechanisms.
    - Too simple to account for many types of accidents, such as those involving suffocation

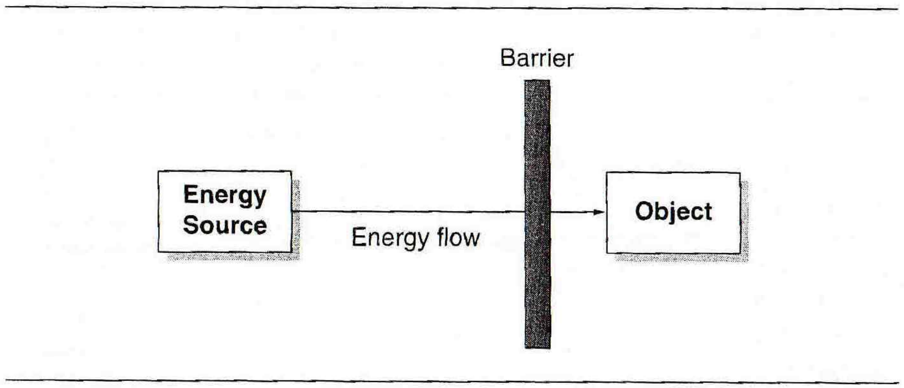


FIGURE 10.1
Simple energy model of accidents.

- MacFarland extended the basic energy model
  - Accidents as resulting from either
    (1) The application of specific forms of energy in amounts exceeding the resistance of the structures on which they impinge
    (2) Interference in the normal exchange of energy between an organism and its environment (including lack of oxygen and exposure to the elements) [140]
  - Sufficient for accidents involving human injury, but not for accidents involving property damage.

- A more general energy model [119] divides accidents into two types:

  (1) Energy transformation accident

  - Occurs when a stable or controlled form of energy is transformed in a way that damages property or injures people.
  - Prevention of such accidents requires controlling the sources or the mechanism of energy transformation or both.

  (2) Energy deficiency accident

  - Occurs when the energy needed to perform a vital function, such as powering the engines in an airplane, is not available and damage to property or injury to human results.
  - Prevention of such accidents requires both
    - A need for some type of energy to perform a function essential for safety (such as brake fluid in an automobile)
    - An event or condition that makes the available energy insufficient to perform the function (such as a leak of brake fluid)

- A final energy model divides systems into two types [5]:
  (1) Those that produce energy (action systems)
    - Hazards are controlled by imposing limitation on the operation of the system.

  (2) Those that constrain energy (nonaction systems)
    - such as pressure vessel, building, and support structure
    - Hazards are controlled by the application of a fixed standard, design allowance, or minimum safety factors.

- In all these energy models, software cannot directly cause an accident.
  - But, it may contribute to the release of energy when it controls or monitors the condition or state of hardware components.

  - Two types of software:
    (1) Command and control software
    (2) Information systems

- Drawback to all of the energy model is that
  - Their scope is limited to energy processes and flows,
  - So may not include accidents involving nonphysical losses caused by logical errors in operation.

  - Therefore, more sophisticated models of '*causal factors*' are needed.

- Causal factors are useful even when the underlying mechanism is energy flow.
  - They expand the possibilities for preventive action, beyond simply controlling the energy directly, to dealing with the conditions and events leading to the loss of energy control.
  - Many categorizations are possible.
    (1) Single event and domino models
    (2) Chain-of-events models
    (3) Models based on systems theory

10.1.2 Domino and Single Events Models

- In the earliest accident models, emphasis shifted from unsafe condition (reflecting industrial safety) to unsafe human acts.
  - Accidents were regarded as someone's fault rather than as an event that could have been prevented by some change in the system.

- Domino Models
  - Heinrich, 1931
  - Hypothesized that people were the cause of accidents.
    - When the first domino falls, it automatically knocks down its neighbor and so on until the injury occurs.
  - Argued that the easiest and most effective domino to remove is the third one.

  - Problems with the domino theory
    - 'ladder example' [261]
    - Might lead to improved inspections procedures, improved training, and a better definition of responsibilities.

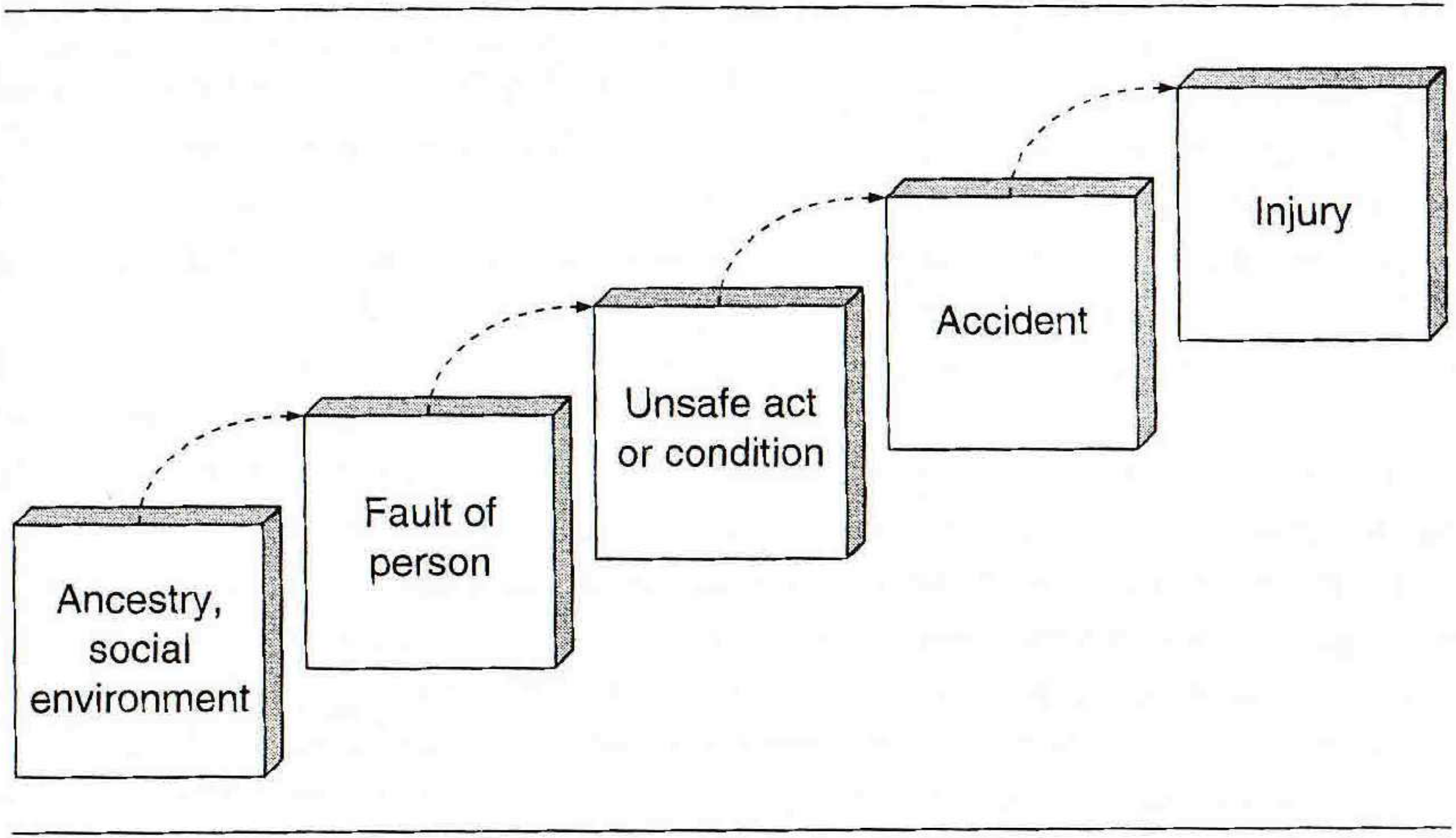


FIGURE 10.2
Heinrich's Domino Model of accidents.

- Bird and Loftus [28] extended the domino theory to include management decisions as a factor in accidents.
  - Lack of control by management, permitting basic causes (personal and job factors) that lead to immediate causes (substandard practices/conditions/errors), which are the proximate cause of an accident or incident, which results in a loss.

- Adams suggested a modified and more general management model [2]
  1. Management structure (objective, organization, and operations)
  2. Operational errors (management or supervisor behavior)
  3. Tactical errors (caused by employee behavior and work conditions)
  4. Accident or incident
  5. Injury or damage to persons or property

- The National Safety Council Model of home accidents, in the 1950s.

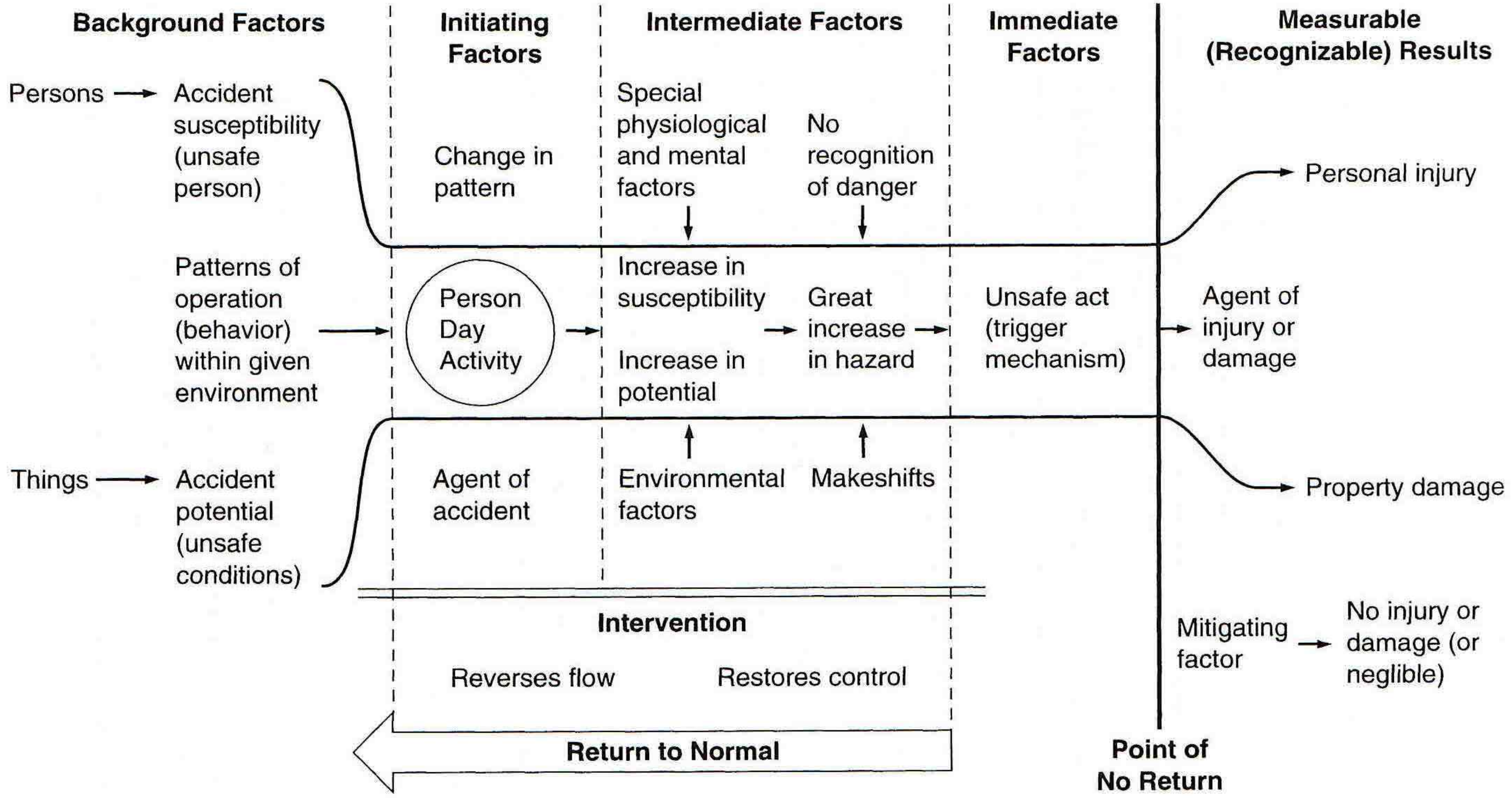


FIGURE 10.3
National Safety Council model of home accidents. (Source: Adapted from National Safety Council. *Journal of Safety Research*, June 1973. Copyright 1973, with kind permission from Elsevier Science Ltd., The Boulevard, Langford Lane, Kidlington OX5 1GB UK.)

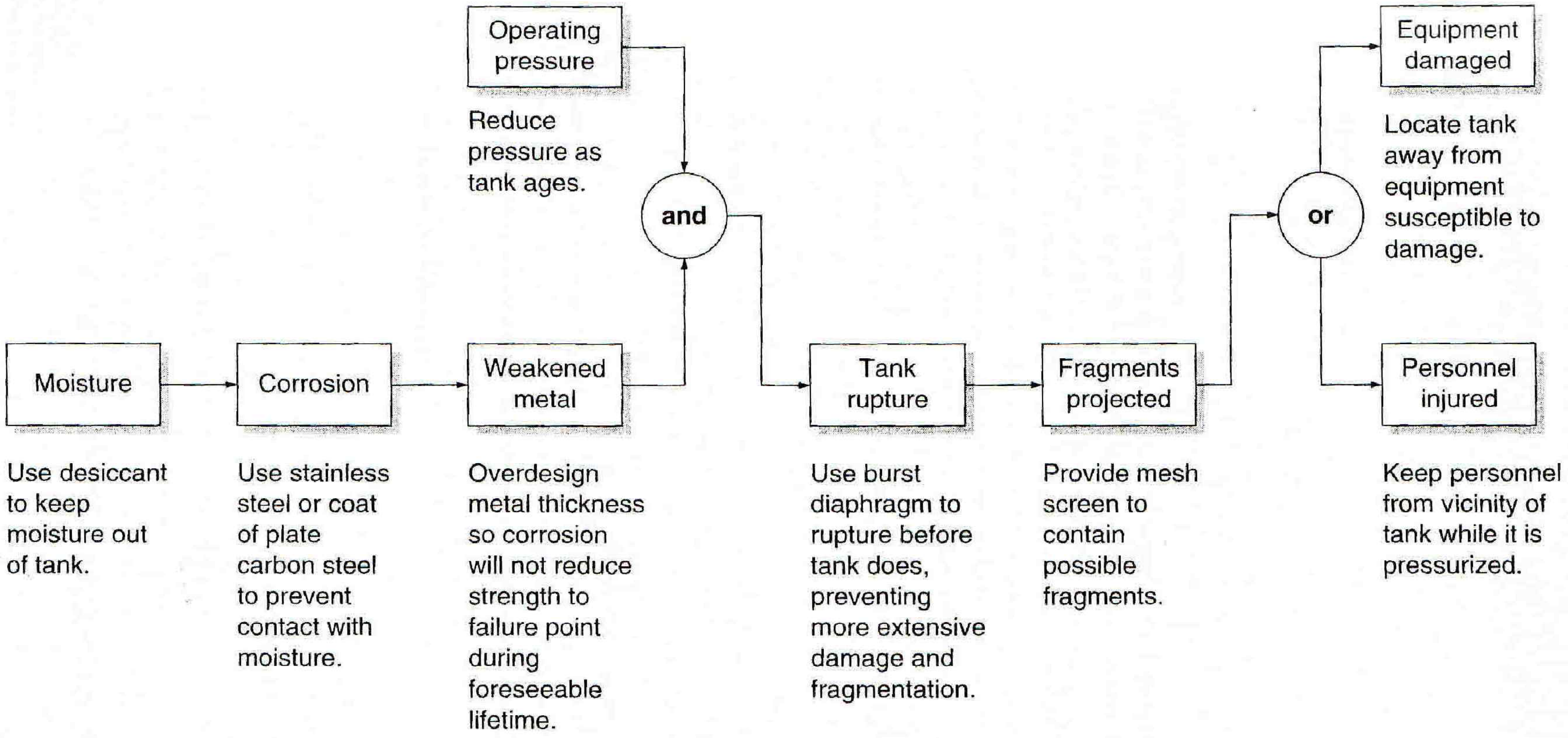
- Epidemiological Models
  - John Gordon, in the 1940s, stresses the multi-factorial nature of accidents.
  - Accidents should be viewed as a public health problem that can be handled using an epidemiological approach.
    - Accidents can be described in terms of the agent (physical energy), the environment, and the host (victim).
    - Accidents are the result of complex and random interactions between these three things.
  - Two types of epidemiology have been applied [338]:
    - Descriptive epidemiology
    - Investigative epidemiology

10.1.3 Chain-of-Events Models

- Models in 10.1.1 and 10.1.2 do not include multiple unsafe acts or conditions, multiple causes of an accident, and multiple causal factors.

- Chain-of-Events model
  - Organizes causal factors into chains of events.
    - Ex) The driver hits the brakes, the car skidded, hits the tree…
  - If the chain can be broken, then the accident will not happen.
  - Prevention measures using this model concentrate on
    - Eliminating particular events or
    - Intervening between the events in the chain

- Simple chain-of-events models have been expanded to include the relationships among events and conditions in accident sequence.
  - Converging chains (logic trees) connected by AND/OR relationships are often used to describe the accident process. (Figure 10.4)



Operating pressure

Reduce pressure as tank ages.

and

Equipment damaged

Locate tank away from equipment susceptible to damage.

or

Moisture

Use desiccant to keep moisture out of tank.

Corrosion

Use stainless steel or coat of plate carbon steel to prevent contact with moisture.

Weakened metal

Overdesign metal thickness so corrosion will not reduce strength to failure point during foreseeable lifetime.

Tank rupture

Use burst diaphragm to rupture before tank does, preventing more extensive damage and fragmentation.

Fragments projected

Provide mesh screen to contain possible fragments.

Personnel injured

Keep personnel from vicinity of tank while it is pressurized.

FIGURE 10.4
A model of the chain of events leading to the rupture of a pressurized tank. (Source: Adapted from Willie Hammer. *Product Safety Management and Engineering*, page 203. Englewood Cliffs, N.J.: Prentice-Hall, 1980. Reprinted with permission of Mrs. W. Hammer.)

- The Perturbation Theory of Accidents
  - Benner [24] describes accidents in terms of a succession of events, where an event is defined as one actor plus one action.
  - Accidents are modeled using parallel horizontal event tracks (with a track for each actor) with cross links to show relationships, all of which are shown above a timeline.

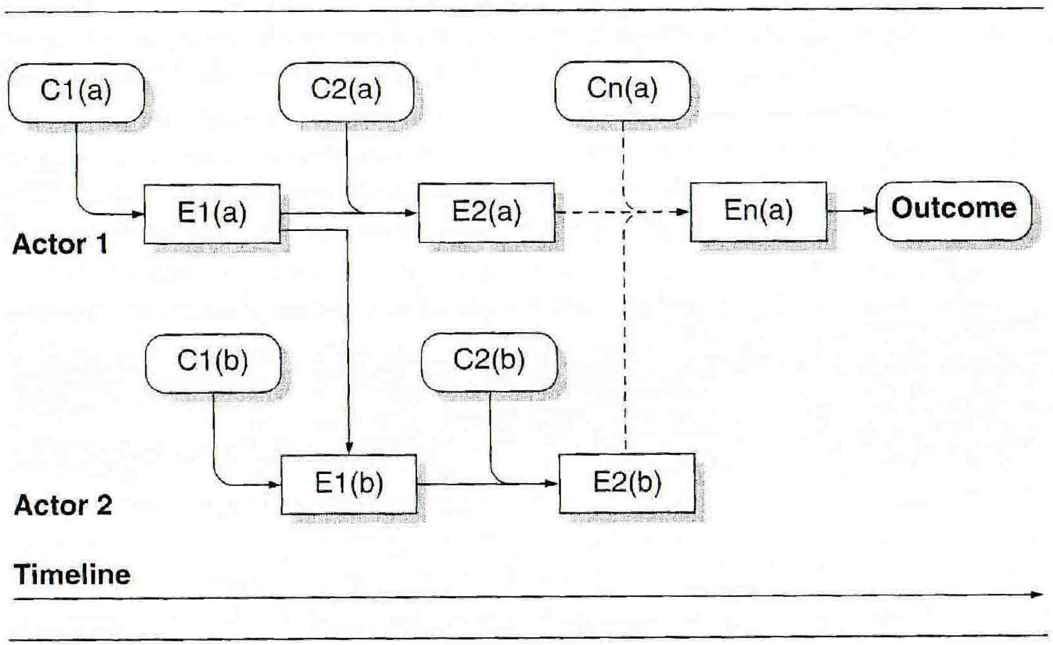
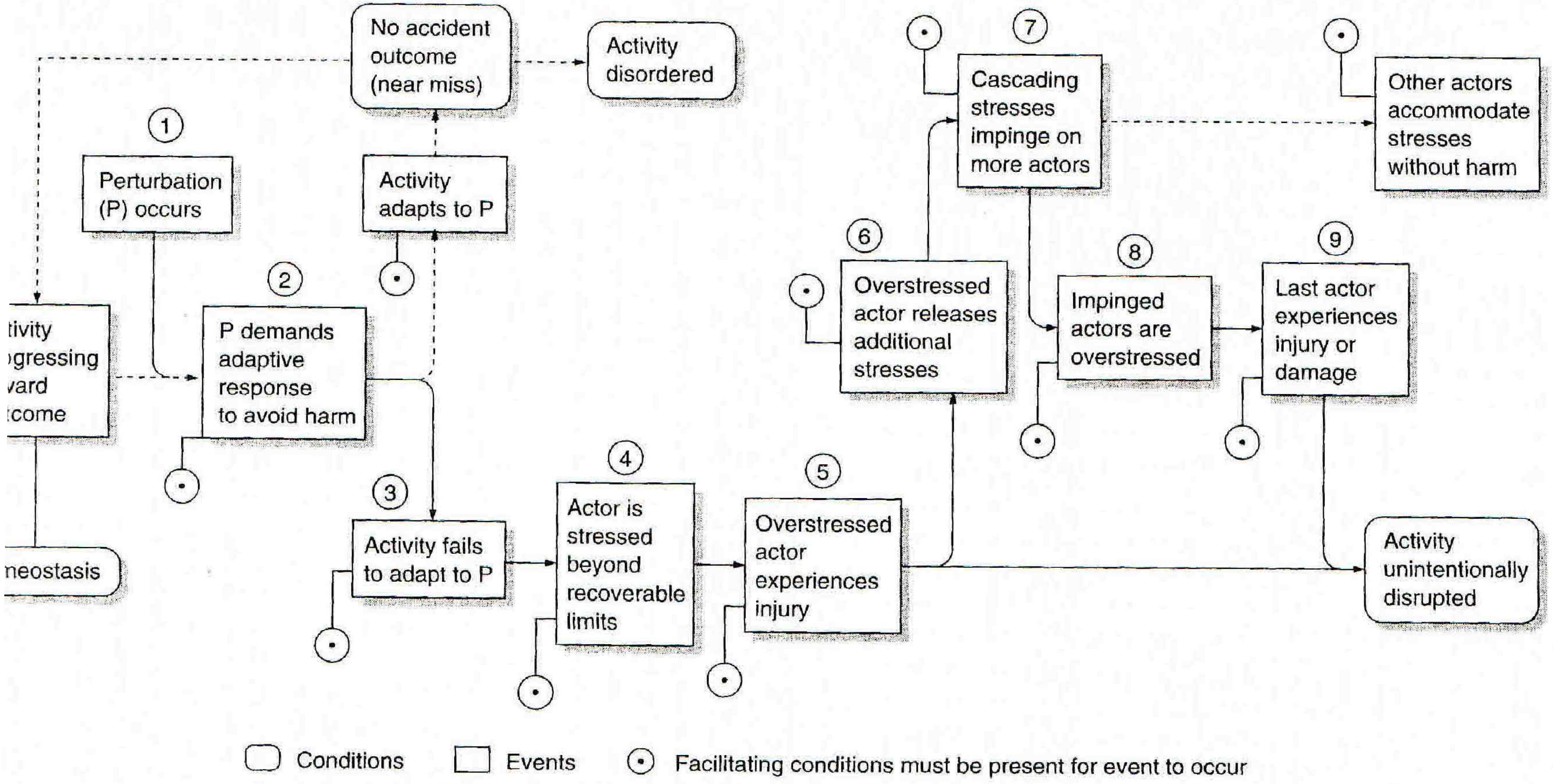


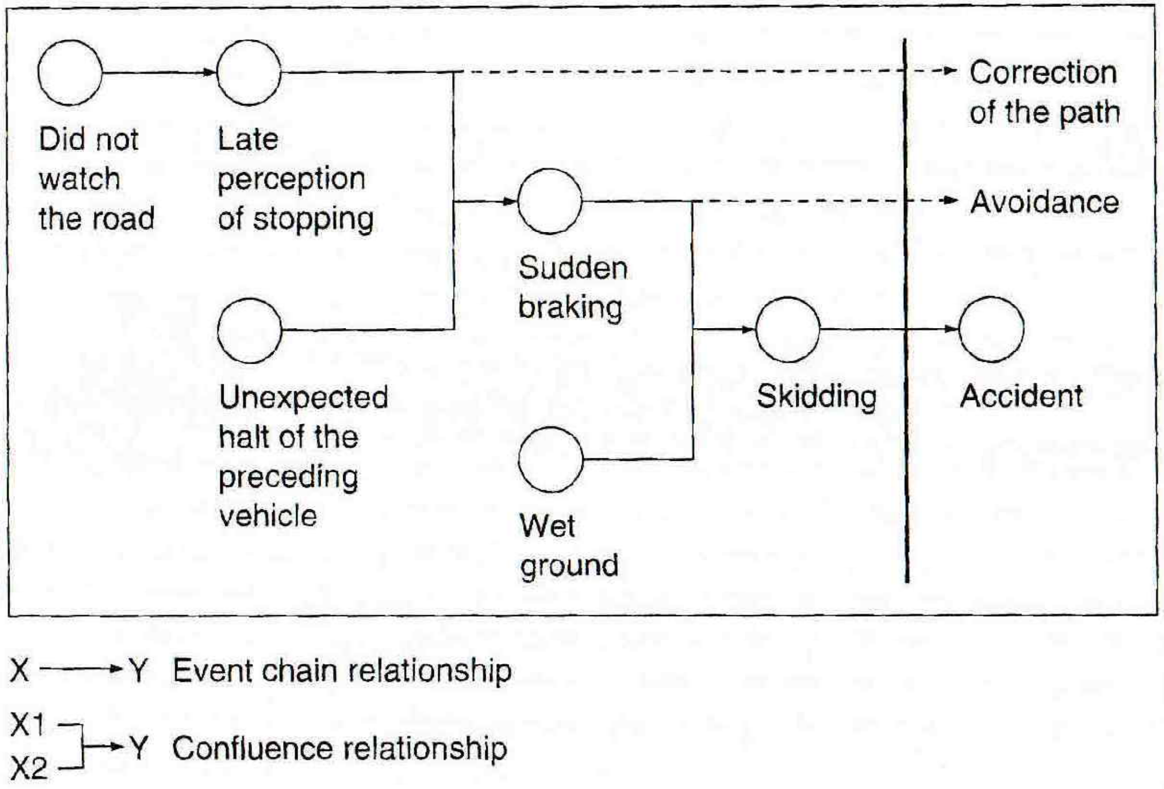
FIGURE 10.5
Multilinear Events Sequence model. (Source: Ludwig Benner, Jr. Accident investigations: Multilinear events sequencing methods. *Journal of Safety Research*, 7(2), June 1975, page 71. Copyright 1975, with kind permission from Elsevier Science Ltd., The Boulevard, Langford Lane, Kidlington, OX5 1GB UK.)

- Perturbation Model
  - Benner calls ‘external influences’ perturbations when they vary from what is usual or expected.
  - If one of the actors fails or is unable to adapt, the perturbation initiates an accident sequence.
  - Figure 10.6
  - … resulting in cascading injury or damage
  - Until the actors are able to accommodate the stress without further harm, the accident continues.
  - If the actor adapt to the perturbation at any time before injury occurs, these will be no accident, even though the activity may be ‘disordered.’


No accident outcome (near miss)
Activity disordered
1
Perturbation (P) occurs
Activity adapts to P
2
tivity gressing vard tcome
P demands adaptive response to avoid harm
3
Activity fails to adapt to P
neostasis
4
Actor is stressed beyond recoverable limits
5
Overstressed actor experiences injury
6
Overstressed actor releases additional stresses
7
Cascading stresses impinge on more actors
Other actors accommodate stresses without harm
8
Impinged actors are overstressed
9
Last actor experiences injury or damage
Activity unintentionally disrupted
Conditions
Events
⊙ Facilitating conditions must be present for event to occur

RE 10.6
Perturbation (P) model of accidents. (Source: Ludwig Benner, Jr. Accident investigations: linear events sequencing methods, *Journal of Safety Research*, 7(2), June 1975, page 69. yright 1975 with kind permission from Elsevier Science Ltd., The Boulevard, Langford Lane, ngton OX5 1GB UK.)

- The INRS Model
  - Deviations or changes are classified in terms of
    - (1) Individual (operator)
    - (2) Machine being operated
    - (3) Surrounding environment
    - (4) Task (the interaction between the operator and the machine)
  - Accidents and incidents are viewed as the consequence of chains of events released or conditioned by a number of variations in normally successful performance. [177]
  - Accidents are mapped as a tree of variations, and points are identified that are sensitive to future improvements (rather than the causes of past events).
    - Emphasis is placed on finding ways to break the accident sequence rather than on removing the cause.


Did not watch the road
Late perception of stopping
Unexpected halt of the preceding vehicle
Sudden braking
Wet ground
Skidding
Accident
Correction of the path
Avoidance
X → Y Event chain relationship
X1, X2 → Y Confluence relationship

(Dashed lines show possible consequences of these actions, if they succeed.)

FIGURE 10.7
An INRS diagram of an accident. (Source: Adapted from Jacques Leplat, Accidents and incidents production: Methods of analysis. In Jens Rasmussen, Keith Duncan, and Jacques Leplat, editors, *New Technology and Human Error*, page 138, New York: John Wiley & Sons, 1987. Reprinted with permission.)

- A problem with most of the chain-of-event models is
  - Factors other than simple events and conditions are difficult to incorporate.
  - Important systematic factors might include structural deficiencies in the organization or factors related to the safety culture in the industry.
- The National Transportation Safety Board Model of Accidents
  - In the 1970s
  - A model and sequencing method
  - Accidents as patterns of direct events and causal factors arising from contributory factors, which in turn arise from systemic factors. (Figure 10.8 and 10.9)
  - Similar to Lewycky's hierarchical model introduced in Chapter 3.

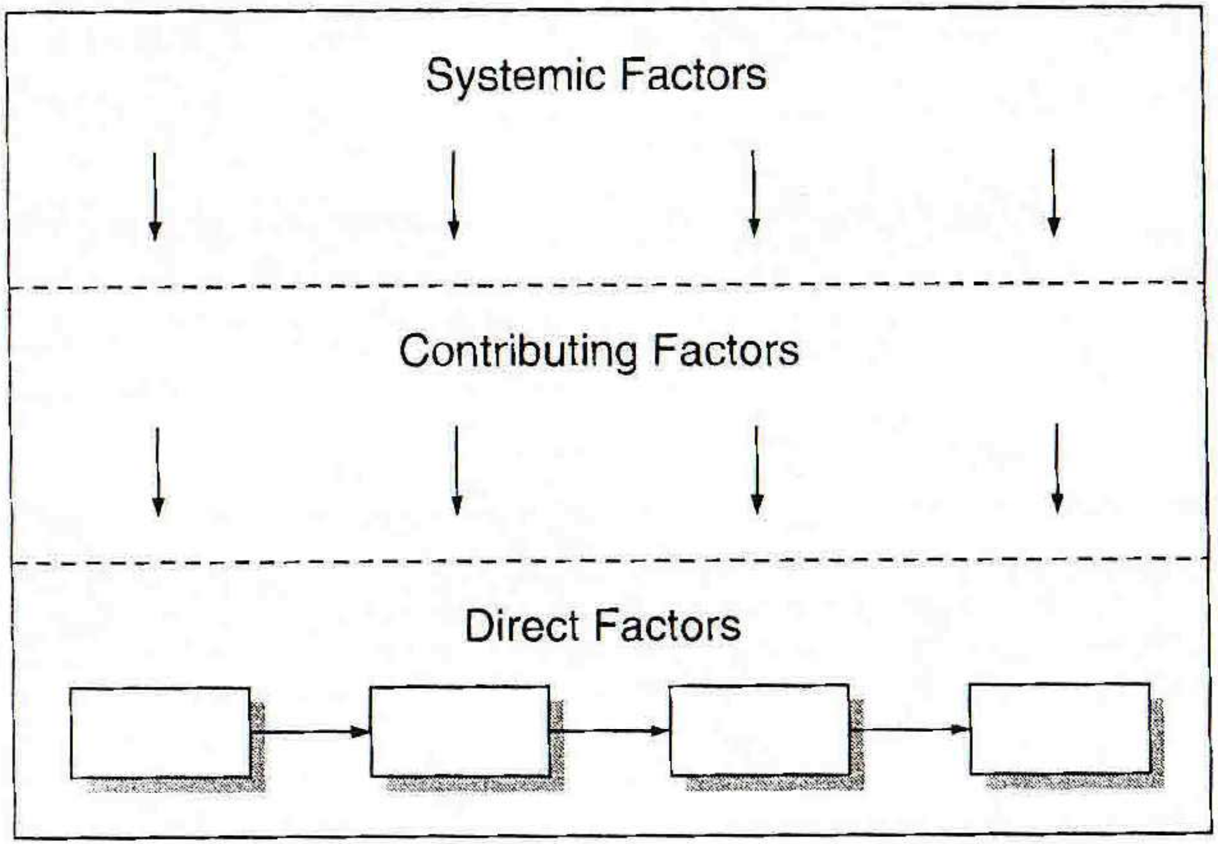


FIGURE 10.8
The National Transportation Safety Board model of accidents.

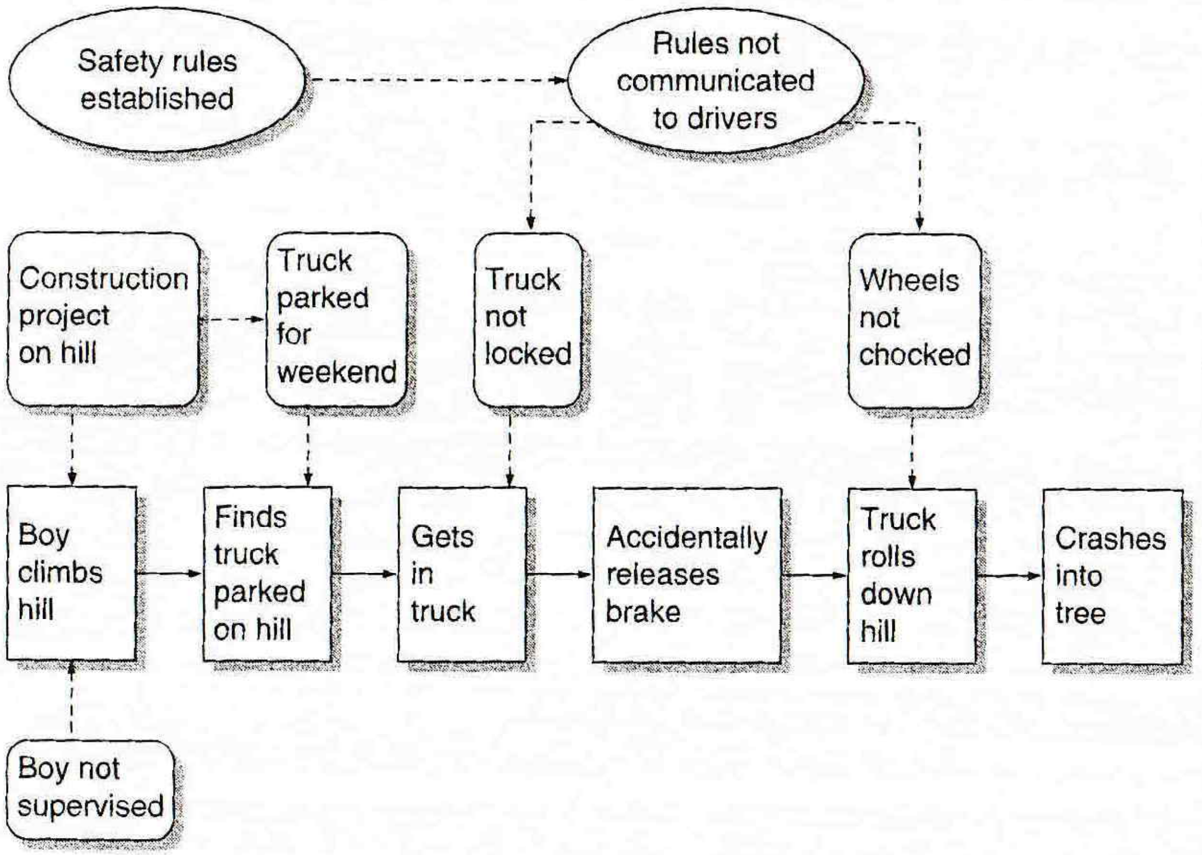


FIGURE 10.9
Example of the National Transportation Safety Board model of accidents. (Source: Adapted from William G. Johnson. *MORT Safety Assurance Systems.* Marcel Dekker, Inc., New York, 1980. Reprinted by permission of Marcel Dekker, Inc.)

- Johnson's MORT Model
  - A model for the Nuclear Regulatory Commission
  - Based on a logic diagram (fault trees) that he calls Management Oversight and Risk Tree (MORT).
  - A very detailed checklist for accident investigation.
    - Over 1,500 basic events or factors are included, and they are related to 98 generic problems.
  - Based on the belief that all accidental losses are caused by unwanted transfers of energy due to a lack of barriers or controls.
    - Accidents result from lengthy sequences of planning and operational errors that fail to adjust to human or environmental changes.
    - Losses arise from two sources
      1. Specific job oversight and omission
      2. Management system that controls the job

- John stress the role of change in accidents, particularly a non-routine operating mode.
  - trials and tests, maintenance and inspection, starting and stopping, etc.

- Because accidents tend to have many causal factors with lengthy sequences of errors and changes,
    - MORT provides a method for breaking down the accident sequence into individual events.

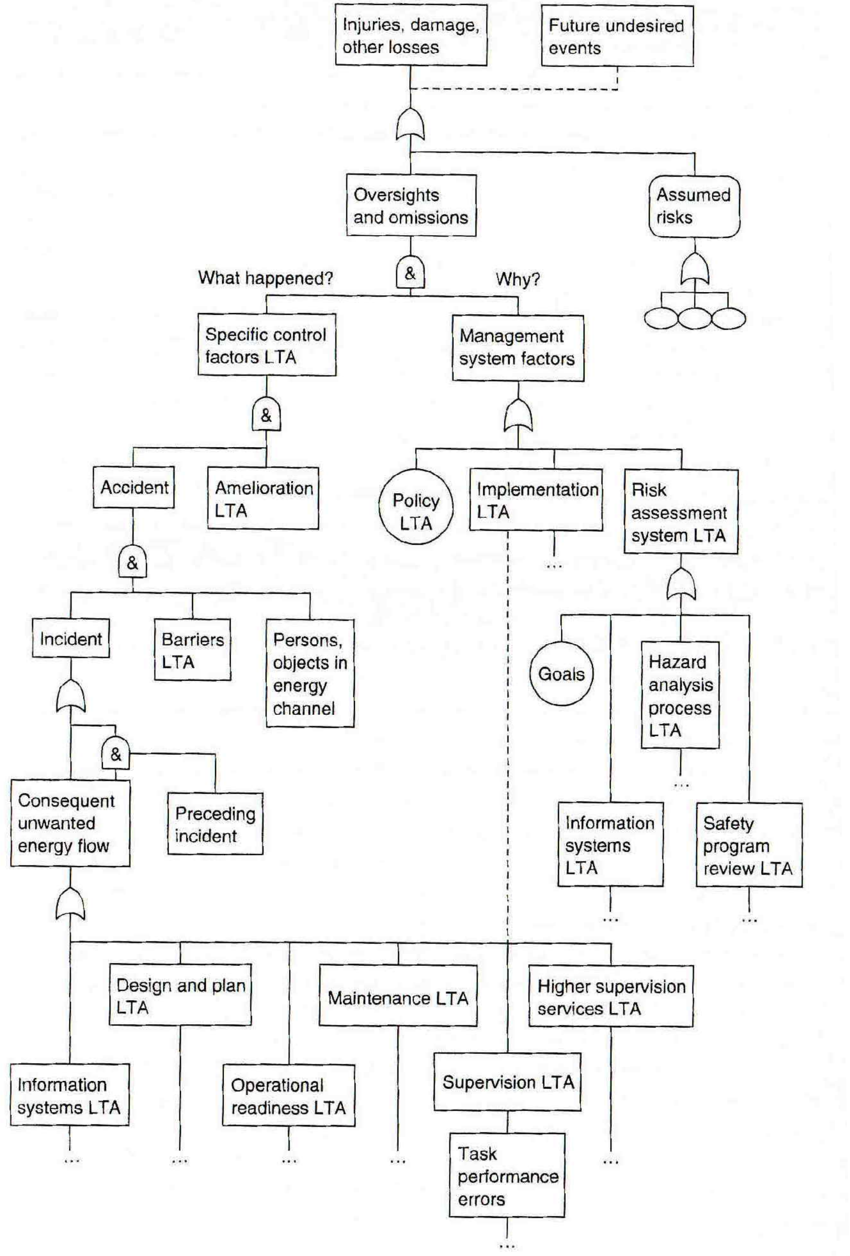


FIGURE 10.10
Top part of the MORT model. Each of the leaf nodes is expanded in further tree structures and each has a checklist associated with it. LTA stands for "less than adequate." (Source: Adapted from William G. Johnson. *MORT Safety Assurance*

10.1.4 Models Based on Systems Theory

- With the rise of system safety, accidents have begun to be viewed in terms of the interactions among humans, machines, and the environment.

- Models based on systems theory consider accidents as arising from the interactions between components of a system.
  - Safety is an emergent property.
    - Arises when the system components interact within an environment.
  - Emergent properties are controlled or enforced by a set of constraints related to the behavior of the system components.
  - Accidents result from interactions among components that violate these constraints.
  - Software often acts as a controller.
    - It embodies or enforces the constraints as it controls the component interactions.
    - It can contribute to an accident by not enforcing the appropriate constraints on behavior or by commanding behavior that violates the constraints.

- A system is merely an abstraction, so different types and levels of subsystems can be identified in these accident models.
  - Industrial engineering models divide the system into appropriate categories of production subsystems.
    1. Johansson
       - Physical, human, information, and management subsystems
    2. [175]
       - Organizational, socio-technical, and elementary human-machine level
    3. [299]
       - Physical, human (production workers), environmental, and managerial subsystems

- Systems models describes accidents in terms of
  - Dysfunctional interactions
  - Control theory
  - Deviations and determining factors

- Dysfunctional Interactions
  - If accidents are seen as arising from the interactions between components or subsystems, they can be eliminated by controlling dysfunctional interactions.
  - Defined as those interactions that lead to hazardous states.
  - Types of dysfunctional interactions (by Leplat [175])
    1. Deficiencies in the articulation of subsystems
       - Three problems types
         1. Boundary areas as zones of insecurity
         2. Zones of overlap as zones of insecurity
         3. Asynchronous evolution of subsystems
    2. Lack of linkup between elements of a system
       - Examples
         1. Poor circulation of information within a group
         2. Lack of correspondence between individual capacity and the task requirements
         3. Incompatibility between the system operation and the mental models/expectations

- Control Theory Models
  - It views safety as a control problem. [274]
  - Systems are viewed as
    - Interrelated components that are kept in a state of dynamic equilibrium by feedback loops of information and control.
  - Accidents occur when
    - Disturbances are not adequately handled by the control system.
  - Rather than focusing on energy flow, it focuses on changes.
    - Can have unforeseen consequences in a complex system, because of incomplete understanding of the system.

- Deviations and Determining Factors
  - [151] emphasizes the role of deviations in accidents.
  - OARU model [323]
  - Deviation
    - Occurs when the value of a system variable falls outside a norm,
    - Where the norm is defined with respect to the planned, expected, or intended production process.
    - May occur in human actions, material, information and instructions, technical equipment, the environment, energy barriers, and relations among activities [150].
  - Accident
    - Occurs as a result of maladaptive response to deviation.
  - Determining factors
    - Relatively stable properties of the production system that vary little over time.
    - Created primarily when the system was established.
  - Deviation, determining factor, or combination of these is necessary for the hazard occurrence. [323]

- The models presented so far handle human error only superficially.
  - Humans play an important role in both causing and preventing accidents.

- The design of a safer human-machine interface requires a better understanding of human behavior.
  - We need to look at cognitive psychology.

10.2 Human Task and Error Models

- Human errors arise in design and operations.
  - To reduce accidents, we need to consider both these sources of error.

- Two ways in which humans contribute to the breakdown of complex systems:
  (1) Active failures
  - Have an immediate adverse effect, such as an operator pushing a wrong button.

  (2) Latent failures
  - May lie dormant for a long time and only become evident when they combine with local triggering conditions.
  - Triggering conditions: active failures, technical faults, or unusual system conditions

- The design of better human-machine interfaces requires that we understand these types of human failures better.
  - Various types of human error have been proposed.

- One simple and common approach
  - Categorize human errors by their external manifestations
    - Omission, commission, not recognizing a hazardous situation
    - Making a wrong decision in response to a problem
    - Responding inadequately to a situation, and poor timing
  - Quickly leads to a combinatorial explosion [274]

- Alternative approaches classify human error according to
  - Task characteristics
  - Human cognitive mechanism
  - Social factors

10.2.1 Task and Environment Models

- Classical human factors theory is concerned with the effects of equipment design, environment, and task structure on the probability of human errors.
  - Considers 'performance-shaping factors'
    - Task structure and workload
    - Physical and psychological stress
    - Design of displays and controls
  - The basic organizing principle behind these models is the task.
    - Tasks can be categorized in two ways:
      (1) by the activity
        - » Coordinating, executing procedures, scanning, recognizing, problem solving, regulating, steering, communicating, planning, recording, and maintaining
        - » One problem with these models: human rarely perform just one task, and performance on a task may be affected by other tasks.

      (2) by the types of human behavior
        - » Simple tasks
        - » Vigilance tasks
        - » Emergency response tasks
        - » Complex tasks
        - » Control tasks

- Tasks categorized by the general types of behavior involved. [172]
  - Simple tasks
    - Composed of relatively uncomplicated sequence of operations involving little decision making.
  - Vigilance tasks
    - Involve the detection of signals.
    - There is a large body of research on this type of task.
  - Emergency response tasks
    - Much less well-defined and their complexity can vary greatly.
    - A common factor in emergency tasks is stress.
    - Two types of behavior are common under emergency conditions [189]:
      - Incredulity response
      - Revert to stereotype
  - Complex tasks
    - Fairly well-defined sequences of operations that involve some decision making
  - Control tasks
    - Include general responsibility for determining whether a control intervention is required and, if so, carrying it out.

- In general, task models provide little help on several dimensions:
  - Accounting for interactions among tasks
  - Determining error modes and probabilities for more complex decision tasks
  - Analyzing or designing new types of tasks and work environments

- They provide only limited help in designing better human-machine interfaces in highly-automated systems being designed today.
  - Simple tasks are being automated.
  - Human are increasingly responsible for the complex tasks.
  - Decision making is not included in these models.

- An alternative is to build models that
  - look beyond the task and environment and incorporate human cognitive processes.

10.2.2 Models Based on Cognitive Mechanisms

- Cognitive models of human error considers the psychological mechanisms used by the operator in performing the task.
  - Interaction of psychological factors with the features of the work environment
  - Human errors are defined in terms of the reasons or underlying psychological mechanisms from which they stem.
  - Explain a large number of behaviors using a small number of psychological mechanisms.
- Many human error models in cognitive psychology.
  - Reason's Model of Human Errors
  - Norman's Model of Slips
  - Mistakes or Human Errors in Planning
  - Human-Task Mismatch

10.2.3 Social Psychology Models

- Human error cannot be thoroughly understood in isolation from the environment in which it occurs.
  - Pure engineering approach (10.2.1) and psychological views (10.2.2) will be inadequate if other social psychology factors are not considered.
  - Reason [278]
    - Acknowledged human error's varied origins in specific conditions of particular environments and in conditions of the human actors.
      - moods, beliefs, values, temperaments
  - Taylor [331]
    - Stressed the importance of factors such as
      - individual value systems
      - feeling of responsibility when trying to understand or account for operator behavior

10.3 Summary

- This chapter has examined a variety of accident models and some well-known human error models.
  - These models of causality are the foundation of approaches to design that attempt to reduce accidents.

Part Four

ELEMENTS OF A SAFEWARE PROGRAM

> *"Accidents are not due to lack of knowledge, but failure to use the knowledge we have."*
>
> *-Trevor Kletz*
> *What Went Wrong?*

Introduction

- Part One
  - Basic understanding of the problem
- Part Two
  - Information about the foundations of system safety
  - How it differs from other approaches to risk reduction
- Part Three
  - Models and definitions upon which to build
- Part Four
  - Examine potential solutions
  - Information about how to implement a comprehensive safety program
    - Enough information about the options available
    - How to make decisions about using these options

Chapter 11

MANAGING SAFETY

Introduction

… Management systems must ensure that there is in being a regime which will preserve the first place of safety in the running of the railway. It is not enough to talk in terms of "absolute safety" and of "zero accident." There must also be proper organization and management to ensure that actions live up to words.

… Sadly, although the sincerity of the beliefs of those in BR [British Rail] at the time of the Clapham Junction accident who uttered such words cannot for a moment be doubted, there was a distressing lack of organization and management on the part of some whose duty it was to put those words into practice. The result was that the true position in relation to safety lagged frighteningly far behind the idealism of the words.

- British Department of Transport
Investigation into the Clapham Junction Railway Accident

Introduction

- System safety engineers have found that
  - The degree of safety achieved is directly dependent upon the emphasis given to it in the organization.

- Management plays a crucial role in determining whether accidents occur.
  - Roles must be carefully defined
    - Organizational structure,
    - Information gathering and documentation procedures, and
    - Specific process to be followed in system development and operation

11.1 The Role of General Management

- The goal of system safety can be achieved only with the support of management.
  - A sincere commitment to safety by management is perhaps the most important factor in achieving it.
  - Three studies [140] found top management's participation in safety issues to be the most effective way to control and reduce accidents.
  - The presence of senior managers, scientists, and engineers with appropriate expertise and knowledge on high-level peer committees is one measure of the quality of and commitment to a safety program.
    - Aerospace Safety Advisory Panel : set up after the Apollo accident in Jan. 1967
  - Other special-purpose and ongoing committees and boards can be used.
    - Education about safety is important.

- In general, management is responsible for
  - Setting safety policy and defining goals
  - Fixing accountability and granting authority
  - Establishing communication channels
  - Setting up a system safety organization

11.1.1 Setting Policy and Defining Goals

- Management can have its greatest influence on safety by
  - Setting safety policy and goals
  - Defining priorities between conflicting goals
  - Establishing procedures for detecting and settling goal conflicts
  - Setting up incentive structures

- A safety policy contains
  - Goals of the safety program
  - A set of criteria for assessing the short- and long-term success of that program with respect to the goals
  - Values to be used in trade-off decisions
  - A clear statement of responsibilities, authorities, accountability, and scope of activities

- Progress in achieving goals should be monitored and improvement identified, prioritized, and implemented. [346]

- Perhaps most important, there must be incentives and reward structures that encourage the proper handling of tradeoffs between safety and other goals.

- When conflicting goals exist, proper tradeoffs can be ensured using two management approaches:
  (1) Establishing strict and detailed guidelines, which sacrifices flexibility
  (2) Leave the decisions to employees, which allows more opportunity for major errors of judgment.

11.1.2 Responsibility, Accountability, and Authority

- Responsibility, accountability, and authority for safety within the organization must be clearly defined by management.
  - Responsibility
  - Authority: the right to command and determine a course of action
  - Accountability: Assessment or measurement of the results of that action
- The three must go together.

- Because of the frequent conflicts between safety and other goals,
  - Independence of the safety group is important.
  - Responsibility for achieving safety should be separate form responsibility for achieving other goals.

- However, in safety-critical, software-intensive programs,
  - It is rare to find anyone in the software development organization with responsibility for the safety of the software in relation to system safety.

11.1.3 Establishing Communication Channels

- Decision makers need information.
  - A set of channels for information dissemination and feedback
  - A means for comparing actual performance with desired performance

- Information must get up to the people who need it.
  - May require redundant channels and cross-checking at successive levels of the organization.
  - Software development groups are often isolated from the main system safety effort.
    - As a result, the software frequently does not reflect system safety concerns and vice versa.
    - Much better two-way communication channels are needed.

11.1.4 Setting up a System Safety Organization

- An effective safety program requires a systematic method and organizational controls.
  - [196] reports that “industrial history is full of examples in which the knowledge to prevent a loss existed but, in the absence of an organization to identify and control hazards, the knowledge was not available to the people who could have avoided the accident.”

- The safety organization must have a high-level, independent role including:
  - Participation in the formulation and implementation of the safety policy
  - Documentation and tracking of hazards and their resolution
  - Education and promotion
  - Adoption and development of standards
  - Conduct of or participation in hazard analysis and other system safety procedures
  - Trend analysis and maintenance of safety documentation
  - Planning and monitoring of testing and operations for safety issues
  - Participation in program reviews and milestones
  - Liaison between other safety groups
  - Accident investigation and analysis

- Personnel Qualifications
  - The system safety job involves extensive meetings, analysis, and negotiation.
    - Communication skills are extremely important.
  - Personnel need to be trained in
    - System engineering techniques
    - Extensive knowledge of and experience with the application involved
    - Quick capacity to learn
  - System safety managers in USA usually are registered as professional engineers and certified as system safety professionals (CSPs).

- Subcontracting
  - In a large program, where subcomponents are contracted to separate companies or organizations,
  - The tasks and scope of the system safety program should be completely specified in the contracting documents.
    - Contractors should be required to develop system and software safety program plans that fit their organizations and products.

- Working Groups
  - In large military programs, system and software safety working groups or committees have proved to be extremely effective in coordinating safety efforts.
  - A functional organization that provides
    - An interface between the agency's safety efforts and its contractors and subcontractors.

- System Safety in Small Organizations
  - Most small organizations cannot afford a full-time safety engineers, and safety responsibility must be assigned to the design engineering department.

  - Even in this situation,
    - Responsibility for safety should be assigned to specific engineers.
    - Written reports and formal presentations should be a part of the product documentation and management review process.
    - Hazard files should be maintained containing all analyses and action taken.

11.2 Place in the Organization Structure

- Organizations are all different.
  - The ideal place for the safety function in each organization will vary.

- Some general requirements are
  - The system safety manager needs a direct link to decision makers.
  - The system safety function needs independence in order to provide an external review divorced from other project management concerns.
  - System safety must have direct communication channels to most parts of the organization. (Figure 11.1)
  - System safety must have influence on decision making.
  - Safety activities must have focus and coordination.

- These requirements suggest some basic principles for locating the safety function within an organization, and some conflicts.
  - System safety manager vs. Program manager

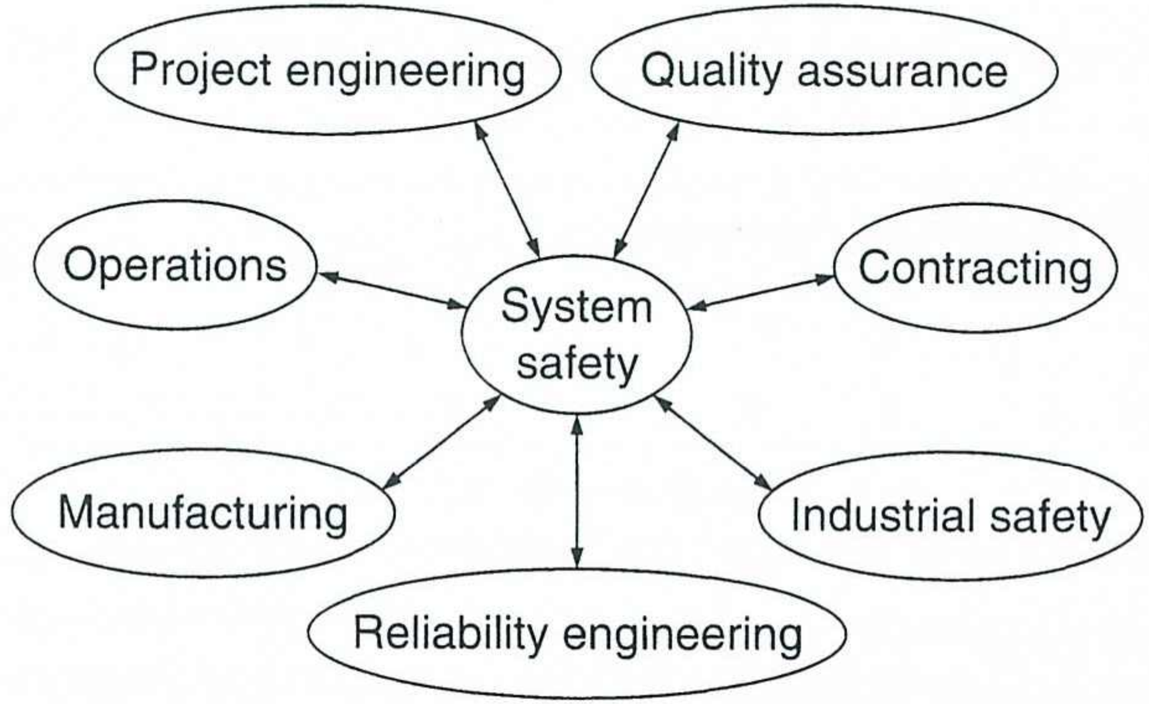


FIGURE 11.1
System safety needs direct communication paths to most parts of the organization.

- Corporate level
  - In a large corporation, system safety at the corporate level can
    - Define and enforce corporate safety policy,
    - Ensure that safe design and review are applied consistently throughout the various decisions and projects.
  - In the U.S. Navy,
    - Weapon System Explosives Safety Review Board
    - Software Systems Safety Technical Review Board
- Lower level
  - At each level below the corporate or headquarters level,
  - A safety group is located at each level, with its function dependent on the level.

- Coordination
  - A coordinating committee or working group is often created to provide coordination and independent channels for safety information.
    - Assures comprehensive and unified planning and action,
    - Allows for independent review.
  - The working groups may be on different levels.

- Small Organizations
  - A small organization may have only one full-time safety engineer or none.
    - Safety responsibility must be assigned to the design engineers.
    - Safety analyses, reports, and presentations still need to be a part of product documentation and management review.
  - Someone needs to be responsible for safety.
  - Someone needs to provide an outside, independent review of the product.
    - Alternatively, an independent organization may contract to provide the safety review.

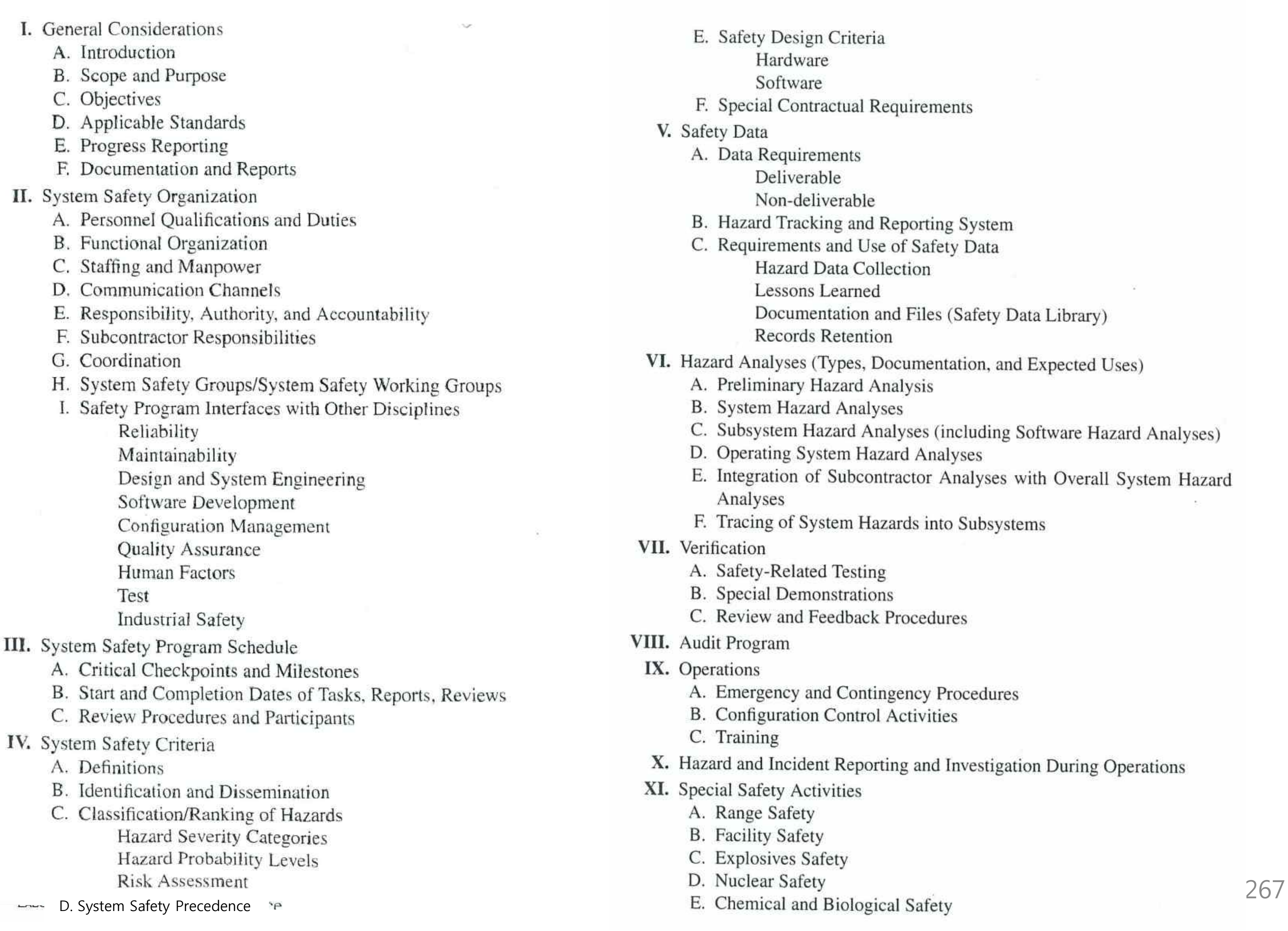
11.3 Documentation

- Three major types of documentation in system safety
  - Planning documents
  - Information systems
  - Reports

11.3.1 Program Plans

- System Safety Program Plan (SSPP)
  - A management document that describes the system safety objectives and how they will be achieved.
  - Provides a regulatory agency, contracting agency, or manager with baseline with which to evaluate compliance and progress.
  - Specifying a plan should be the first step in any safety-critical project.
  - Plans for subsystem safety should be part of the SSPP.
- Many standards exist for program plans.
  - Each is a little different, but all contain similar information.
  - Devising a general plan for everyone is not practical.
  - Each plane needs to be tailored to its project and goals and to fit the corporate personality and management system.

- The following information might be included in a comprehensive plan:



I. General Considerations
   - A. Introduction
   - B. Scope and Purpose
   - C. Objectives
   - D. Applicable Standards
   - E. Progress Reporting
   - F. Documentation and Reports

II. System Safety Organization
   - A. Personnel Qualifications and Duties
   - B. Functional Organization
   - C. Staffing and Manpower
   - D. Communication Channels
   - E. Responsibility, Authority, and Accountability
   - F. Subcontractor Responsibilities
   - G. Coordination
   - H. System Safety Groups/System Safety Working Groups
   - I. Safety Program Interfaces with Other Disciplines
     - Reliability
     - Maintainability
     - Design and System Engineering
     - Software Development
     - Configuration Management
     - Quality Assurance
     - Human Factors
     - Test
     - Industrial Safety

III. System Safety Program Schedule
   - A. Critical Checkpoints and Milestones
   - B. Start and Completion Dates of Tasks, Reports, Reviews
   - C. Review Procedures and Participants

IV. System Safety Criteria
   - A. Definitions
   - B. Identification and Dissemination
   - C. Classification/Ranking of Hazards
     - Hazard Severity Categories
     - Hazard Probability Levels
     - Risk Assessment
   - E. Safety Design Criteria
     - Hardware
     - Software
   - F. Special Contractual Requirements

V. Safety Data
   - A. Data Requirements
     - Deliverable
     - Non-deliverable
   - B. Hazard Tracking and Reporting System
   - C. Requirements and Use of Safety Data
     - Hazard Data Collection
     - Lessons Learned
     - Documentation and Files (Safety Data Library)
     - Records Retention

VI. Hazard Analyses (Types, Documentation, and Expected Uses)
   - A. Preliminary Hazard Analysis
   - B. System Hazard Analyses
   - C. Subsystem Hazard Analyses (including Software Hazard Analyses)
   - D. Operating System Hazard Analyses
   - E. Integration of Subcontractor Analyses with Overall System Hazard Analyses
   - F. Tracing of System Hazards into Subsystems

VII. Verification
   - A. Safety-Related Testing
   - B. Special Demonstrations
   - C. Review and Feedback Procedures

VIII. Audit Program

IX. Operations
   - A. Emergency and Contingency Procedures
   - B. Configuration Control Activities
   - C. Training

X. Hazard and Incident Reporting and Investigation During Operations

XI. Special Safety Activities
   - A. Range Safety
   - B. Facility Safety
   - C. Explosives Safety
   - D. Nuclear Safety
   - E. Chemical and Biological Safety

11.3.2 Safety Information System

- Information system is crucial to the success of safety efforts.
  - Although setting up a comprehensive and usable information system can be time consuming and costly.
  - Resources invested in it will be well spent.

- An organization's system safety information system will contain
  - Updated System Safety Program Plan
  - Status of the activities included
  - Results of hazard analyses
  - Tracking and status information on all known hazards
  - Incident and accident information including corrective action
  - Trend analysis data
  - Well-defined interfaces with various project databases
  - And so on.

- Feedback of information on which to base future decisions
  - A crucial aspect of any management system
  - Information can be used to describe to
    (1) Diagnose
    (2) Compare
    (3) Evaluate
    (4) Improve

- Understanding its limitations and inaccuracies is important.
  - Simply collecting information is not enough.
  - The information must be accurate and timely, and must be disseminated to the appropriate people in a useful form.
    - Information collection
    - Analysis
    - Dissemination

- Collection
  - Various types of information can be collected.
    - Performance and operational data
    - Accident and incident investigations
    - Results of studies and evaluations
    - Technical information such as standards, manuals, and professional literatures
    - Operational data and so on.
  - Very expensive and requires a large quality control group working over an extensive period of time.
  - Data may be seriously distorted by the way it is collected.
    - Systematic filtering or suppression
      - Data from near-miss reporting by operators are often filtered.
    - Unreliability
      - Difficult to maintain reliable reporting of incidents over an extended period. [151]
      - Therefore, this data is not useful.

- Various research studies have shown that it is possible to improve the comprehensiveness and reliability of data collection [151].
  - Anonymous reporting, directive reporting, and reporting during a limited time period.
    - Improved near-miss reporting.
  - Critical Incident Technique
    - Shown that there is 1 accident for approximately every 400 near misses [108].
  - Model accident analysis forms
    - Encourage the inclusion of multiple factors and events.
  - Automating monitoring
    - Can greatly improve the accuracy and completeness of data collection

- Analysis
  - Data, once collected, need to be analyzed and summarized.
  - Systematizing and consolidating a large mass of data into a form useful for learning is difficult.
    - Raw quantitative data can be misleading and should always be tested for statistical significance.
    - Statistical analysis too can be misleading and can leave out important information for hazard control.
  - Analysis is at the heart of any system safety program.
    - It will be covered in depth in Chapters 13, 14, and 15.

- Dissemination
  - The most difficult aspect of maintaining an information system
  - Information should be presented to decision makers in a meaningful way.
  - Traditionally, information has been disseminated in
    - Checklists
    - Standards
    - Codes of practice
  - Information about hazards has been distributed in manuals,
    - Dow Index in the chemistry industry (Chapter 14)
    - Books by Trevor Kletz on the chemical industry [157, 161]
  - Accident/incident files are one of the most important information sources for hazard analysis and control.
    - The information needs to be presented in a form that people can learn from, apply to their daily jobs, and use throughout the life cycle of projects.

- An Example Information System: The ASRS
  - The Air Safety Reporting System (ASRS)
  - In 1975, USA
- CHIRP
  - Confidential Human Factors Incident Reporting Program
  - In Britain
- EUCARE
  - The European Confidential Safety Reporting Network
  - In Europe

11.3.3 Safety Reports

- Although every organization or industry has different names for reports, there are actually a few types:
  - Hazard reports
  - Design documentation
  - Hazard analysis reports
    - On the procedures used and the results
  - Safety assessment reports
    - Integrates all system, hardware, and software analyses
  - Final safety assessment reports
    - At the end of a project
    - Used
      - To determine compliance with the program safety requirements
      - To provide system users and operators with a comprehensive description of the system hazards and hazardous subsystems and operations associated with the system.

- Once the management system is in place, the next step is to define a process tailored for the particular project and its participants.

Chapter 12

THE SYSTEM AND SOFTWARE SAFETY PROCESS

Introduction

For us, the indisputable lesson of Chernobyl lies in this: the principles regulating the further development of the scientific-technological revolution must be safety, discipline, order, and organization. Everywhere and in all respects, we must operate according to the strictest standards.

- Mikhail Gorbachev

- Each project is different.
  - System safety processes and tasks need to be tailored.

- In this chapter, a generic process is outlined that may be
  - Mapped onto existing standards, and
  - Used to design specific processes for new standards of for specific projects and companies.

12.1 The General Tasks

- System safety can be thought of as an integrating function – it ensures
  - Specific safety considerations are introduced into the program early in the life cycle.
  - They are integrated into overall system design and development.
  - Efforts continue throughout the system's existence.

- The tasks involved in system safety differ in the various phases of a project, but continue throughout the system life cycle.

- An engineering project can be divided into five stages:
  (1) Conceptual development
  (2) Design (demonstration and validation)
  (3) Full-scale development
  (4) Production and deployment (pre-startup)
  (5) Operation

12.1.1 Conceptual Development Tasks

- At this early stage of system development, essential groundwork must be established for the entire project safety effort.
  - Develop the system safety program plan.
  - Establish the safety information and documentation files.
  - Establish the hazard auditing and log file.
    - Develop a tracking system within configuration control (both system and subsystem, including software) for tracing hazards and their resolution.
  - Review lessons learned and applicable documents.
  - Establish certification and training requirements for personnel involved in the development, test and operation of the system.
  - Participate in system concept formation.
  - Delineate the scope of the analyses.
  - Identify hazards and safety requirements.
  - Identify design, analysis, and verification requirements, possible safety-interface, problems, including the human-machine interface, and operating support requirements.
  - Establish working groups and other special structures.

12.1.2 System Design Tasks

- System design tasks
  - Update analyses.
  - Participate in system tradeoff studies.
  - Ensure that safety requirements are incorporated into system and subsystem specifications, including human-machine interface requirements.
  - Ensure that identified hazards are being eliminated or controlled in the evolved design.
  - Identify safety-critical components.
  - Trace system hazards into components.
    - Identify subsystem (including software) safety requirements and constraints on subsystem functionality, design, and verification. Identify the parts of the software that control safety critical operations.
  - Review general test and evaluation procedures.
    - Start planning system and subsystem safety testing and evaluation procedures. Develop test plans, test descriptions, test procedures, and test case requirements.
  - Review training and operations plans.
  - Evaluate design changes for safety impact.
  - Document all safety decisions and maintain safety information.

12.1.3 Full-Scale Development Tasks

- Full-Scale Development Tasks
  - Review and update the hazard analyses. Perform interface analyses, including interfaces within subsystems (such as between safety-critical and non-safety-critical software components).
  - Ensure that safety requirements and constraints are incorporated into the subsystem (including software) safety requirements and constraints.
  - Ensure that the software engineers and code developers understand software-related system safety requirements and constraints.
  - Trace software safety requirements and constraints through to the code.
    - Identify safety-critical software components and variables to the code developers.
  - Review engineering designs and design documents.
    - Ensure that safety requirements and constraints are incorporated into subsystem designs and documentation.
  - Update testing and verification requirements.
  - Review both system and subsystem test results.
    - Trace back results to system hazards.
  - Evaluate changes for their impact on safety.
  - Design and evaluate training and operational procedures.
  - Perform final evaluation of product design.

12.1.4 System Production and Deployment Tasks

- System Production and Deployment Tasks
  - Update hazard analyses.
  - Perform safety evaluation and verification at the system and subsystem levels.
  - Perform safety inspections.
  - Ensure that safety-related information is incorporated into user and maintenance documents.
  - Review change proposals for safety impact.
  - Perform a final evaluation of the produced and deployed system.

12.1.5 System Operation Tasks

- System Operation Tasks
  - Update procedures.
  - Maintain an information feedback system.
  - Review and analyze accidents and incidents.
  - Conduct safety audits.
  - Review changes and maintenance procedures.

12.2 Examples

- Each process must be tailored to a particular system and organization.
  - But, examples of processes and organizational structures can be useful in understanding the options available.

- Three examples
  - (1) An European underground rail station
    - No computer were involved, but software considerations could be easily be incorporated within the same overall process.
  - (2) A Combat Weapon System
  - (3) The NASA Space Shuttle Project
    - Because of their larger scale, required of a much different and more elaborate organizational structure.

12.2.1 An Underground Rail Station

- An underground rail station in Zurich [361]
  - An excellent example of a well-planned system safety process for a relatively small project in Europe
  - Complex structure
  - It had to deal with the interdisciplinary nature of its structure.

- The safety process had to be flexible throughout the various life cycle phases and accommodate changing needs. (Figure 12.1)

- Definition of Scope
  - Before starting the preliminary hazard analysis, the safety personnel carefully defined its scope, considering the information and time available and the possible results.
  - Once the scope defined, the safety personnel documented the information to be used such as drawings and findings from ongoing research.

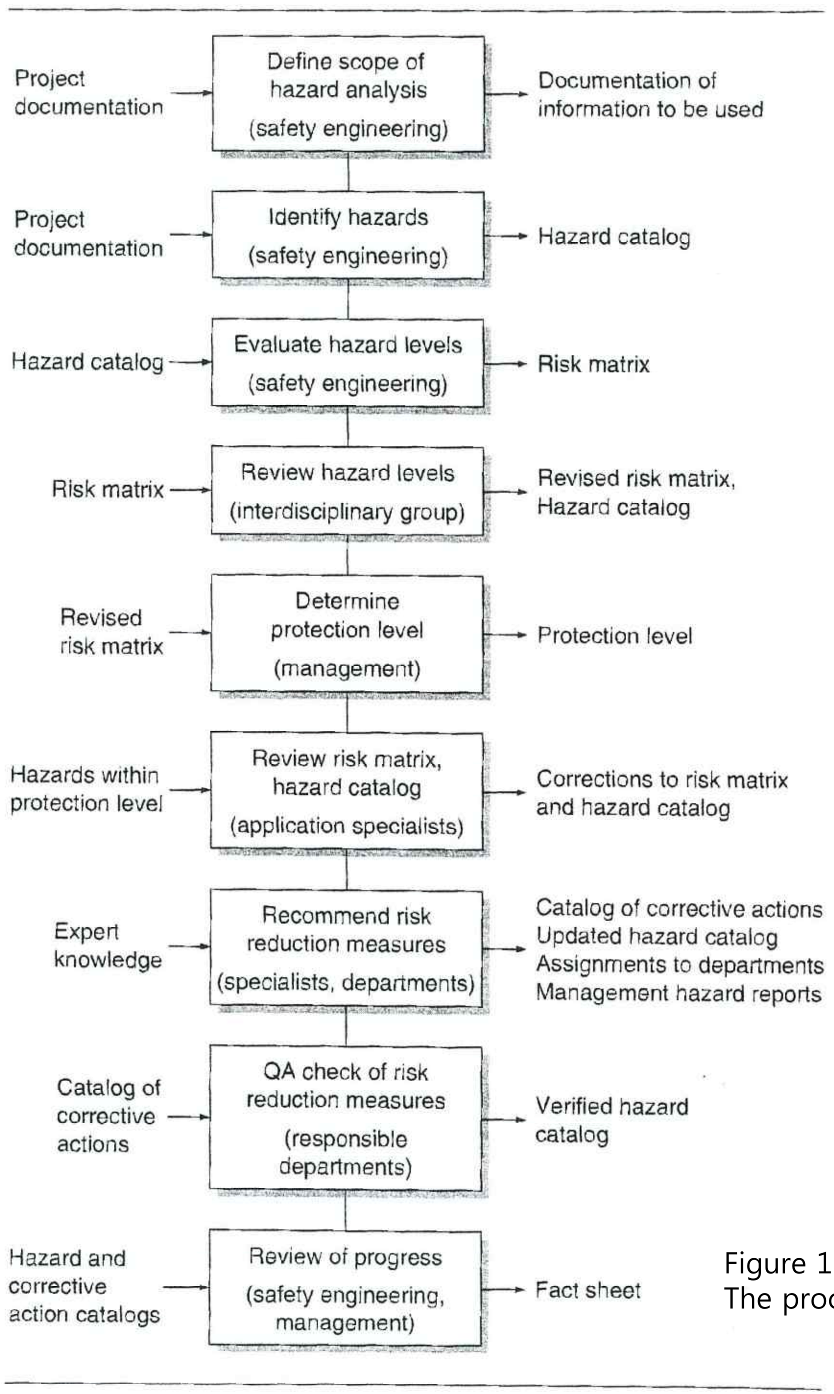
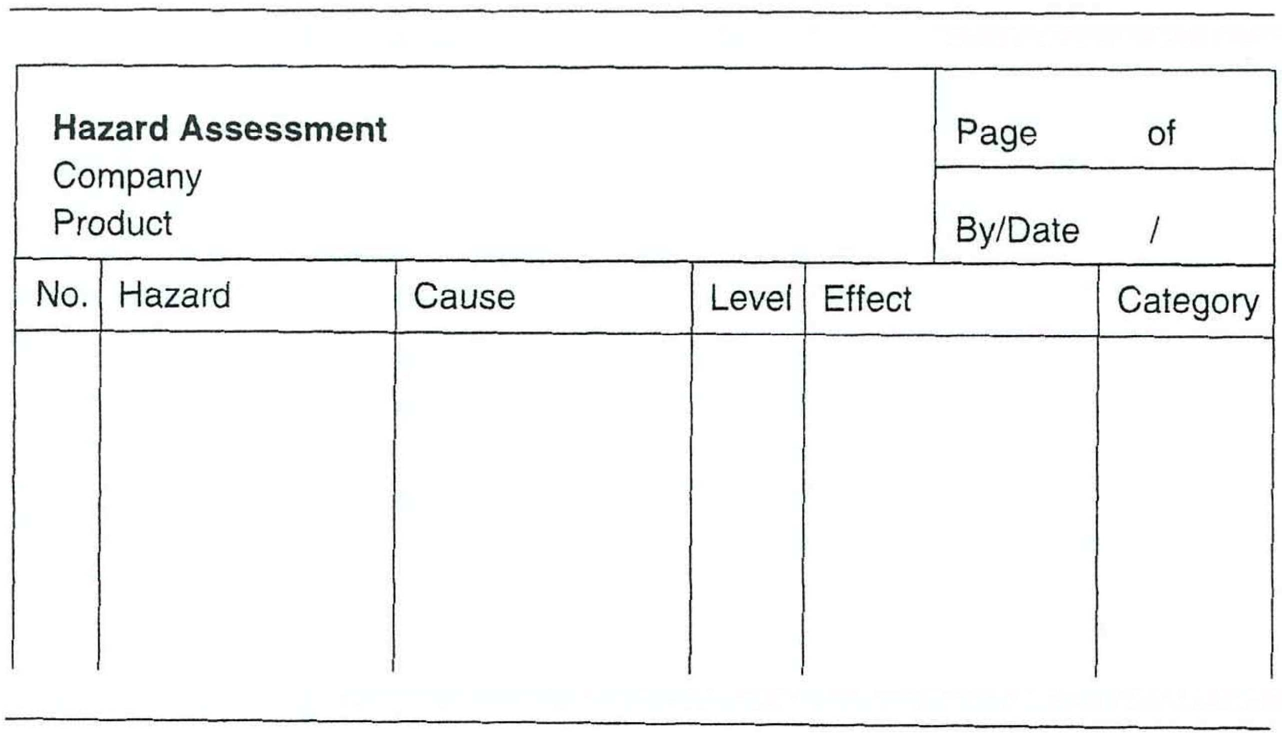

Project documentation
Define scope of hazard analysis (safety engineering)
Documentation of information to be used
Project documentation
Identify hazards (safety engineering)
Hazard catalog
Hazard catalog
Evaluate hazard levels (safety engineering)
Risk matrix
Risk matrix
Review hazard levels (interdisciplinary group)
Revised risk matrix, Hazard catalog
Revised risk matrix
Determine protection level (management)
Protection level
Hazards within protection level
Review risk matrix, hazard catalog (application specialists)
Corrections to risk matrix and hazard catalog
Expert knowledge
Recommend risk reduction measures (specialists, departments)
Catalog of corrective actions
Updated hazard catalog
Assignments to departments
Management hazard reports
Catalog of corrective actions
QA check of risk reduction measures (responsible departments)
Verified hazard catalog
Hazard and corrective action catalogs
Review of progress (safety engineering, management)
Fact sheet

Figure 12.1
The process steps in the underground rail project

- Hazard Identification and Assessment
  - The initial hazard identification
    - Included the hazard, level, effect, and category in a fairly standard way.
    - Done by a small group of safety engineers without consulting project specialists.

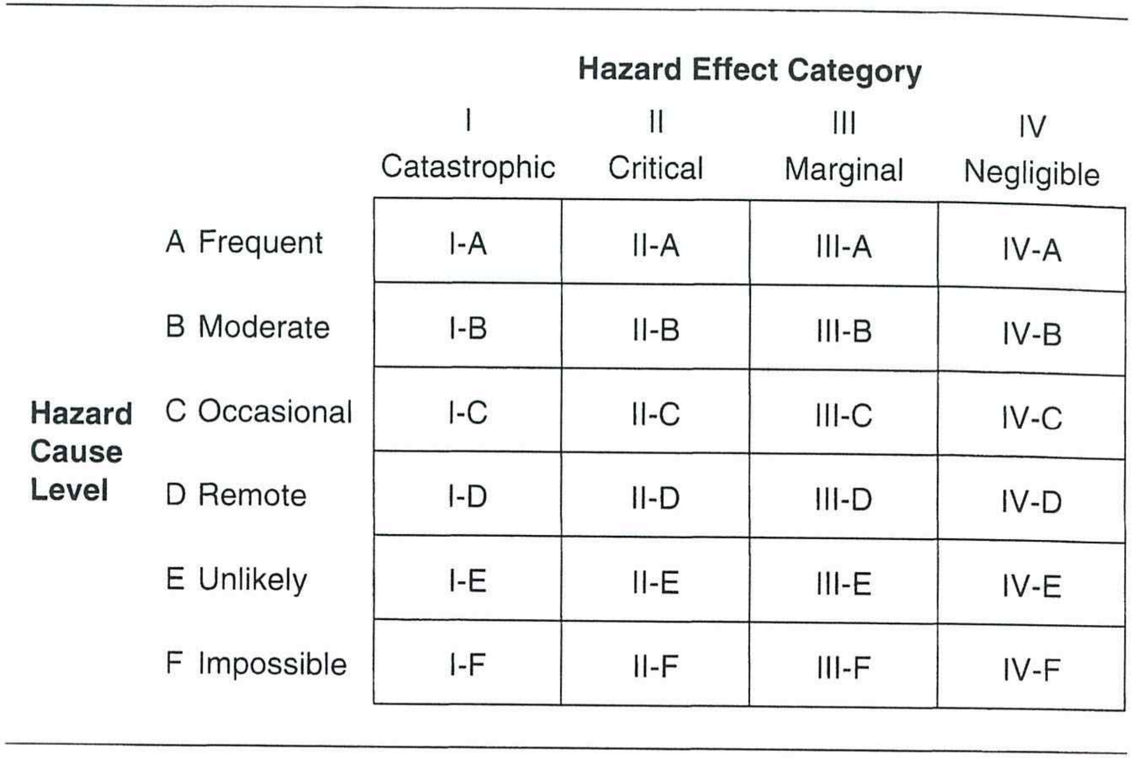


Hazard Assessment Company Product	Page of By/Date /

No.	Hazard	Cause	Level	Effect	Category

FIGURE 12.2
Hazard identification form.

- Next step
  - Assign a hazard level from among six levels presenting the relative probability of the occurrence of a potential cause.
  - The hazard level was determined by using a standard matrix (Figure 12.3) to combine hazard severity and hazard probability.



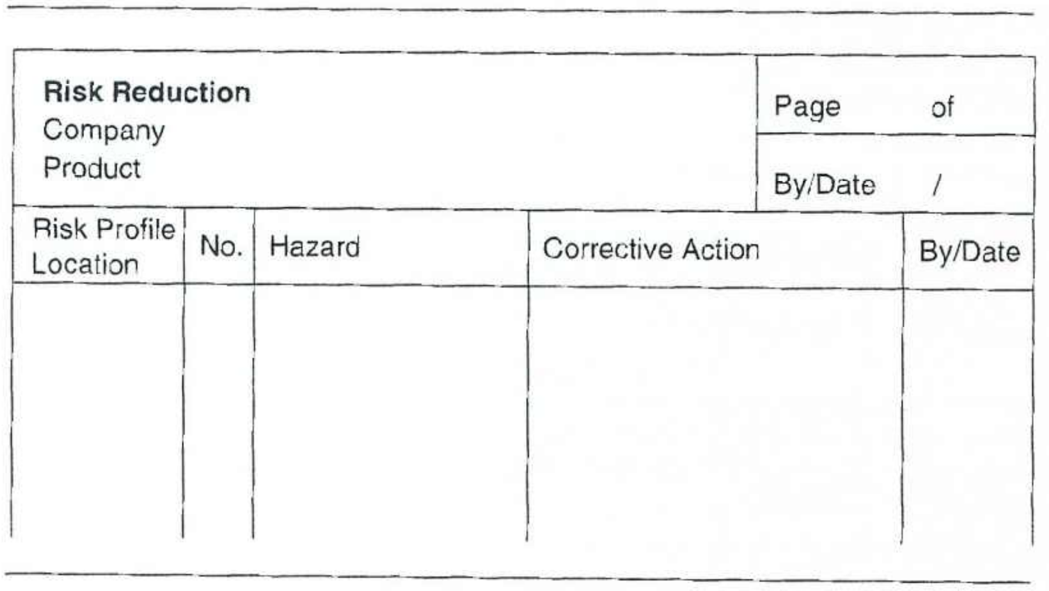
		Hazard Effect Category			
		I	II	III	IV
		Catastrophic	Critical	Marginal	Negligible
Hazard Cause Level	A Frequent	I-A	II-A	III-A	IV-A
	B Moderate	I-B	II-B	III-B	IV-B
	C Occasional	I-C	II-C	III-C	IV-C
	D Remote	I-D	II-D	III-D	IV-D
	E Unlikely	I-E	II-E	III-E	IV-E
	F Impossible	I-F	II-F	III-F	IV-F

FIGURE 12.3
A typical hazard level matrix.

- The protection level
  - Determined at a meeting of management by examining the current protection level in other part of the public transportation system.

- With the additional information about protection level added to the risk matrix, the complete hazard catalog and risk matrix were sent to the members of the specialist group to double check the assessment of all hazards that appeared below the established protection level.

- Risk Reduction
  - The standard system safety precedence (Chapter 8) was used for risk reduction.
  - Information about each hazard above the protection level was documented, and then the hazard were assigned to the specialists or departments that might be able to contribute to the risk reduction.
  - Recommended risk reduction measures were collected and assembled into a ‘catalog’ of corrective actions. (Figure 12.4)



Risk Reduction Company Product				Page of By/Date /
Risk Profile Location	No.	Hazard	Corrective Action	By/Date

FIGURE 12.4
Risk reduction form.

- Risk Reduction
  - A quality assurance check of the updated catalog was performed by asking all involved departments to double check their assignment.
    - The reassessment was completed after their corrective actions were provided.
  - Each corrective action taken was crossed off in the catalog and the person or department responsible for this action noted.

- The hazard catalog, risk profile, and risk reduction catalog provided high visibility of safety issues throughout the project. [361]

12.2.2 A Combat Weapon System

- A much larger project
  - The much greater scale makes the interaction more complex and difficult to implement.

- A major difference stems from the fact that
  - The customers (the U.S. Navy and NASA) play a larger role in the development and safety process.

  - Defense system are developing using a safety standard, MIL-STD-882.
  - The Navy has a project-independent board.
    - Navy Weapons System Explosives Safety Review Board (WSESRB)
    - Software System Safety Technical Review Board
    - Navy Safety Study Group
    - Software Safety Working Group (SSWG)

- A U.S Navy cruiser and destroyer combat system
  - Built in 1980s, and includes nuclear weapons.

- Nuclear Safety Advisory Group (NSAG)
  - Systems that include nuclear weapons are subject to different and more stringent standards than are nonnuclear systems.
  - Supported the nuclear weapons safety studies and evaluation performed by the Safety Study Group (SSG).

- The prime contractor led the safety engineering efforts.
  - In this company, System Safety Engineering (SSE) is a part of system engineering.
  - SSE reviewed specifications and generated safety products early.
    - PHA (Preliminary Hazard Analysis)
    - PHA was then distributed to the design engineering groups and to other engineering disciplines (reliability, maintainability, human factors engineering)
  - Throughout the program, the prime contractor’s SSE group participated in component and system design reviews.

- Tasks
  - The overall system safety efforts consists of three major functions:
    1. (1) Establishment of a safety baseline
    2. (2) Identification and elimination or control of hazards
    3. (3) Safety verification

- Establishment of a safety baseline
  - Evaluating the safety of new design features using design reviews and equipment safety histories in Navy safety data files.
  - SSE
    - Prepared the PHA.
    - Prepared and evaluated hazard reports.
    - Defined the safety requirements for the specifications.
    - Started the system hazard analysis.

- Identification and elimination or control of hazards
  - Used various analyses to track identified hazards through elimination, control, or procedural precautions.
  - SSE
    - Updated the PHA.
    - Continued the system hazard analysis.
    - Participated in the SSWG and NSAG activities.
    - Performed various other types of required hazard analyses.
    - Compared the Safety Summary Report (SSR)
- Safety verification
  - Includes a combination of inspections, documentations, tests, and data analyses.
  - To determine whether all system components could be operated and maintained without risk to personnel or equipment.

- Type of Hazard Analyses
  - A relatively large number of hazard analyses were performed on this program:
    - Preliminary Hazard Analysis (PHA)
    - System Hazard Analysis (SHA)
    - Operating Hazard Analysis (OHA)
    - Maintenance Hazard Analysis (MHA)
    - Computer Program Safety Analysis (CPSA)
    - Subsystem Hazard Analysis (SSHA)
    - Radiation Hazard Analysis (RHA)
    - Nuclear Safety Analysis (NSA)
    - Inadvertent Launch Analysis (ILA)
    - Weapon Control Interface Analysis (WCIA)

- Safety Criteria
  - Hazard probability ranking

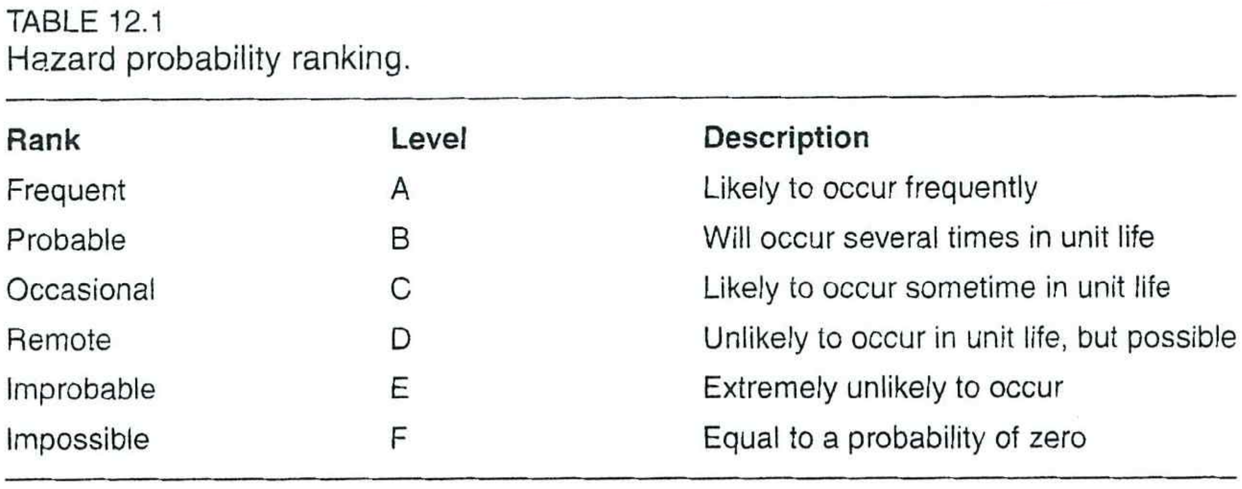
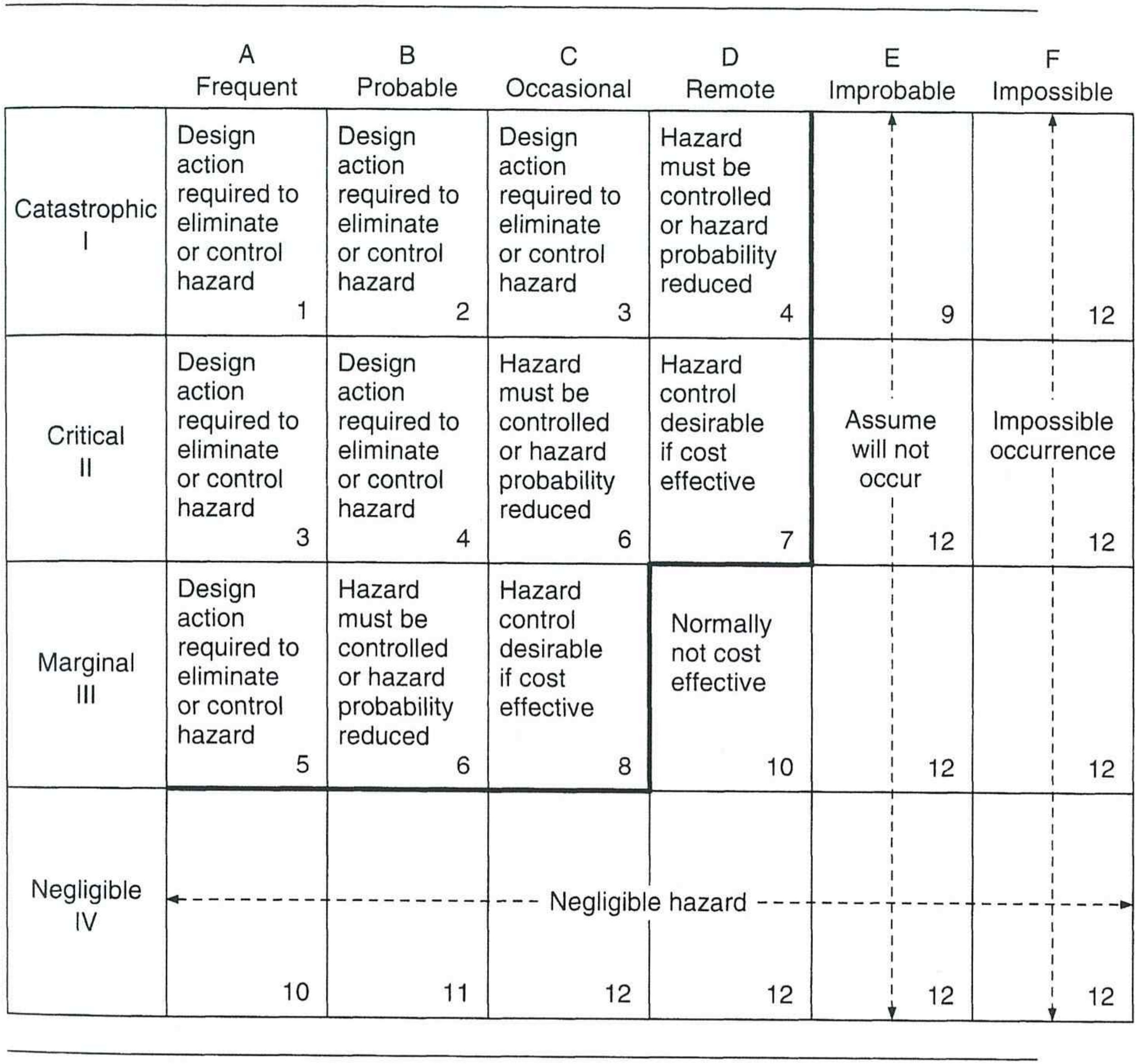


TABLE 12.1
Hazard probability ranking.

Rank	Level	Description
Frequent	A	Likely to occur frequently
Probable	B	Will occur several times in unit life
Occasional	C	Likely to occur sometime in unit life
Remote	D	Unlikely to occur in unit life, but possible
Improbable	E	Extremely unlikely to occur
Impossible	F	Equal to a probability of zero

  - Hazard criticality index matrix were used to combine hazard severity and hazard probability. (Figure 12.5)



	A Frequent	B Probable	C Occasional	D Remote	E Improbable	F Impossible
Catastrophic I	Design action required to eliminate or control hazard 1	Design action required to eliminate or control hazard 2	Design action required to eliminate or control hazard 3	Hazard must be controlled or hazard probability reduced 4	9	12
Critical II	Design action required to eliminate or control hazard 3	Design action required to eliminate or control hazard 4	Hazard must be controlled or hazard probability reduced 6	Hazard control desirable if cost effective 7	Assume will not occur 12	Impossible occurrence 12
Marginal III	Design action required to eliminate or control hazard 5	Hazard must be controlled or hazard probability reduced 6	Hazard control desirable if cost effective 8	Normally not cost effective 10	12	12
Negligible IV	Negligible hazard 10	11	12	12	12	12

FIGURE 12.5
Hazard criticality index matrix.

- Reports
  - The reports used for this system included
    - Safety test reports
    - System hazard alert reports
    - A general safety analysis summary report
    - A combat system safety statement
    - A nuclear safety analysis report
    - Various non-deliverable reports, memoranda, and analyses

- Safety Hazard Reporting Procedures
  - Anyone observing or encountering an erroneous condition in his opinion, was required to report it to the prime contractor SSE office within one working day and to document any immediate action.
  - Figure 12.6 shows the controls and procedures for handling safety hazard and alerter reports.

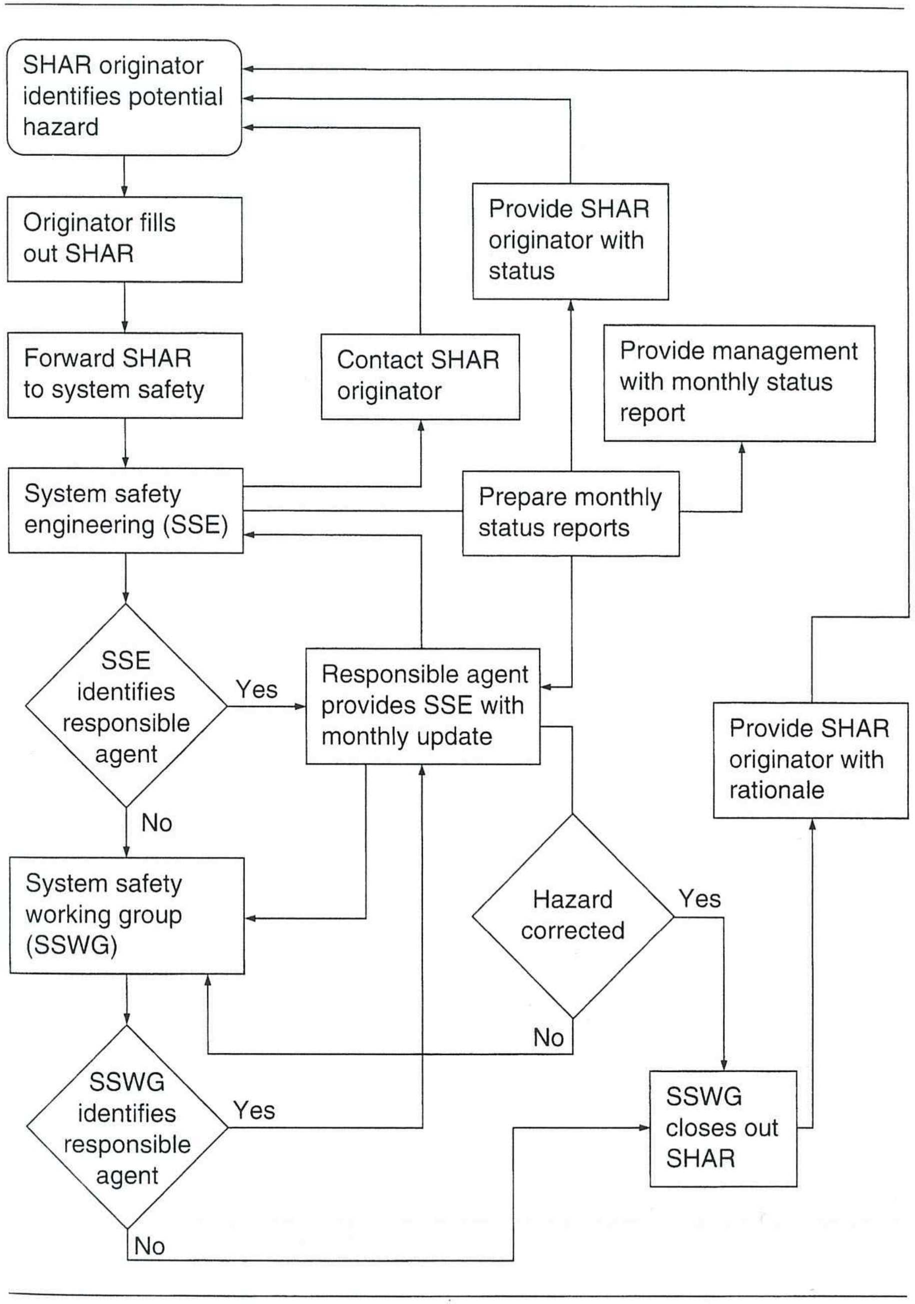

SHAR originator identifies potential hazard
Originator fills out SHAR
Forward SHAR to system safety
System safety engineering (SSE)
Contact SHAR originator
Provide SHAR originator with status
Provide management with monthly status report
Prepare monthly status reports
SSE identifies responsible agent
Yes
No
Responsible agent provides SSE with monthly update
Provide SHAR originator with rationale
System safety working group (SSWG)
Hazard corrected
Yes
No
SSWG identifies responsible agent
Yes
No
SSWG closes out SHAR

FIGURE 12.6
Safety hazard correction procedure.

12.2.3 The NASA Space Shuttle Project

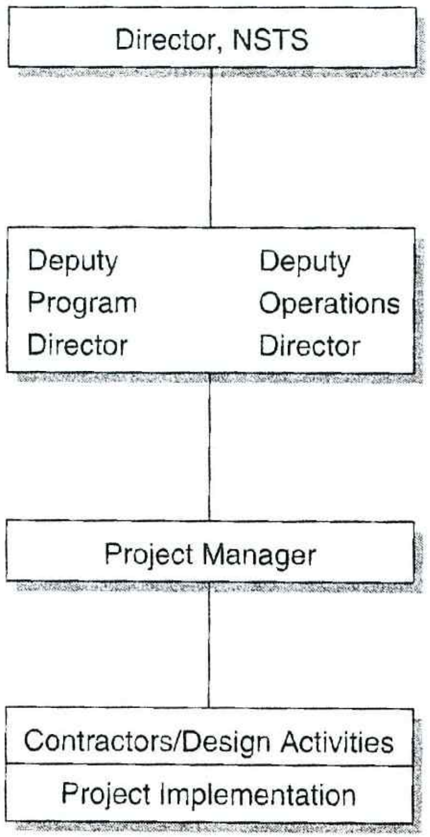
- The Challenger accident
  - An example of process that did not work well
  - However, may be a valuable or more valuable than those that have not experienced problems.

- Criticisms of the process that surfaces during the investigation of the Challenger accident are included.
  - Along with changes that were made to the process after the accident and more recent evaluations of current practices.
  - The Space Shuttle is one of the most complex engineering projects.
    - NASA should be commended for its open attitude toward public examination of its process.
    - It is from this type of openness that we learn and improve.

- Emphasis in the following is on the process being followed in the operational phase of the National Space Transportation System (NSTS) program.

- Overall NASA Safety Policy
  1. Avoid loss of life, injury of personnel, damage, and property loss.
  2. Instill a safety awareness in all NASA employees and contractors.
  3. Assure that an organized and systematic approach is utilized to identify safety hazards and that safety is fully considered from conception to completion of all agency activities.
  4. Review and evaluate plans, systems, and activities related to establishing and meeting safety requirements both by contractors and by NASA installations to ensure that desired objectives are effectively achieved.

- Management Structure
  - Prior to the Challenger accident
    - The NSTS Program was managed out of the Johnson Space Center (JSC) in Huston.
  - After the accident
    - The NSTS Program Director was brought to NASA headquarters.
  - Each management level has one or more associated boards or panels (Figure 12.8).
  - Review groups associated with the safety processes (Figure 12.9).



LEVEL I Top-level program requirements. Budgets and schedules. Control of changes above $1 million per year or two million total or those impacting level-I requirements or schedules.

LEVEL II Management and integration of all elements of the program. Integrated flight and ground system requirements, schedules, and budgets; control of project interfaces, control of changes exceeding project budgets, or those impacting level-II requirements, interfaces, or schedules.

LEVEL III Project-oriented flight and ground system requirements, schedules, and budgets; control of changes within project level budgets, schedules, and specifications.

LEVEL IV Detailed flight and ground system requirements within assigned project. Control and implementation of detailed design.

FIGURE 12.7
National Space Transportation System program management relationships. (Source: Reprinted with permission from *Post-Challenger Evaluation of Space Shuttle Risk Assessment Management*, 1988. Courtesy of the National Academy Press. Washington, D.C.)

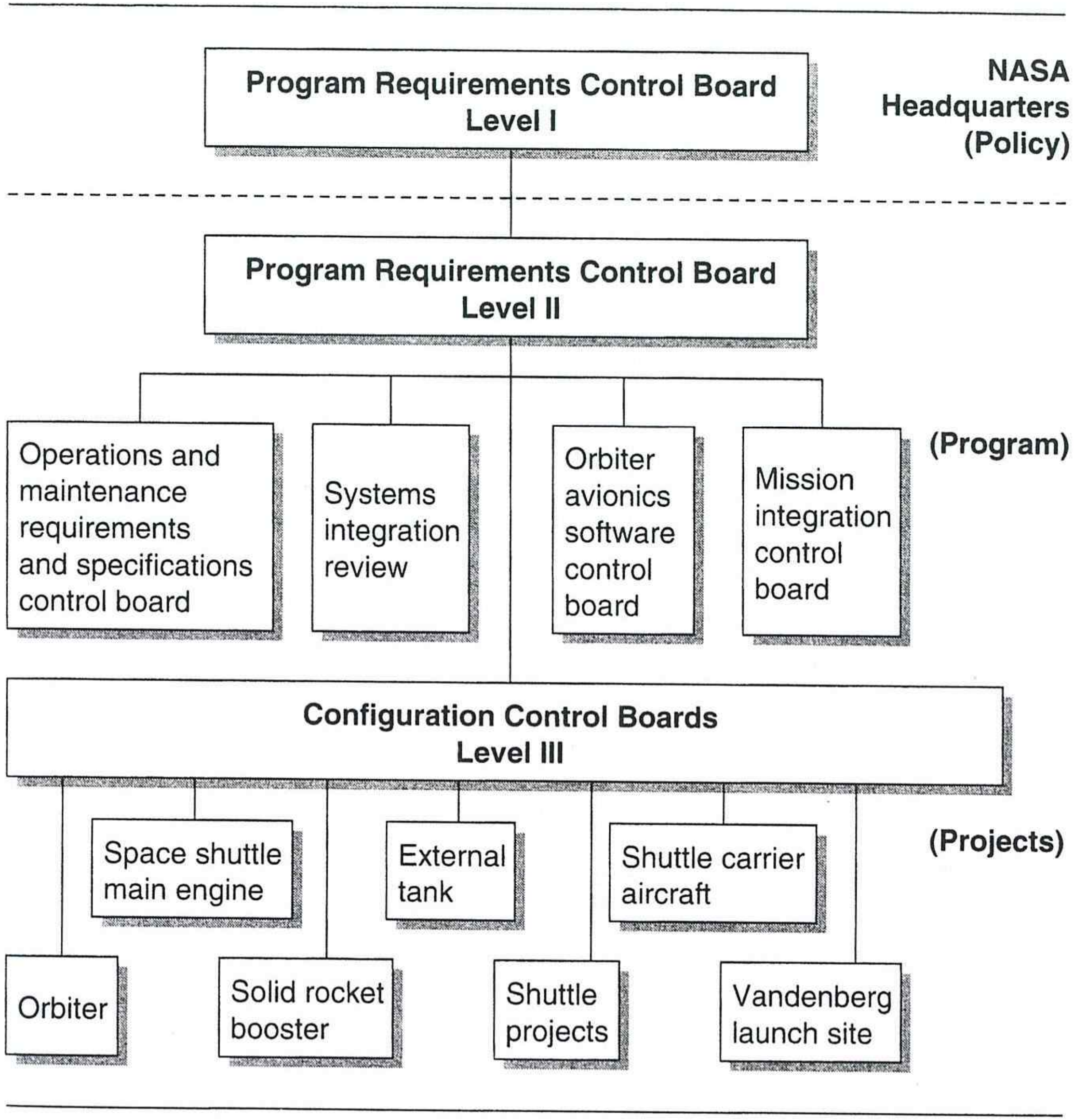

Program Requirements Control Board
Level I
NASA
Headquarters
(Policy)
Program Requirements Control Board
Level II
Operations and maintenance requirements and specifications control board
Systems integration review
Orbiter avionics software control board
Mission integration control board
(Program)
Configuration Control Boards
Level III
Space shuttle main engine
External tank
Shuttle carrier aircraft
(Projects)
Orbiter
Solid rocket booster
Shuttle projects
Vandenberg launch site

FIGURE 12.8
Structure of NSTS Program Requirements Control Boards and Configuration Control Boards. (Source: Reprinted with permission from *Post-Challenger Evaluation of Space Shuttle Risk Assessment Management*, 1988. Courtesy of the National Academy Press. Washington, D.C.)

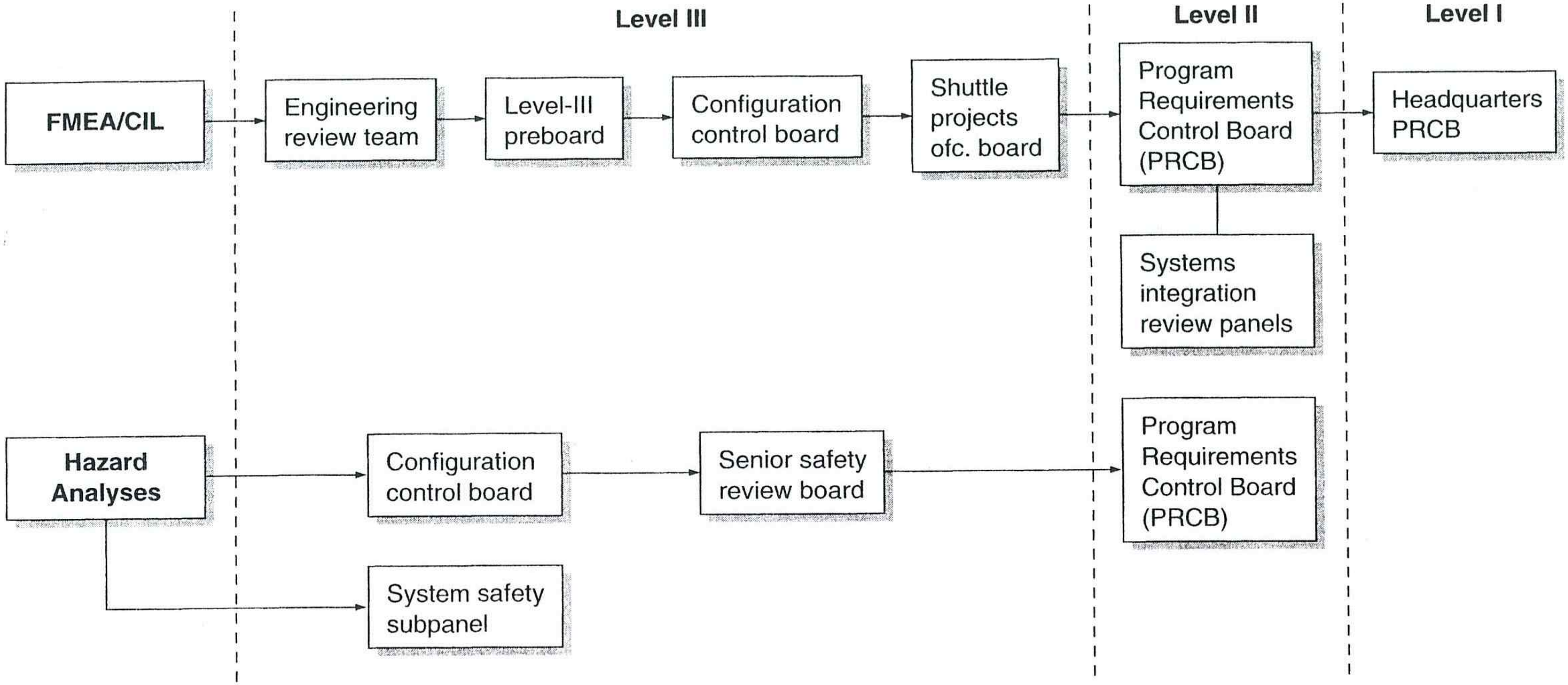

Level III
Level II
Level I
FMEA/CIL
Engineering review team
Level-III preboard
Configuration control board
Shuttle projects ofc. board
Program Requirements Control Board (PRCB)
Headquarters PRCB
Systems integration review panels
Hazard Analyses
Configuration control board
Senior safety review board
Program Requirements Control Board (PRCB)
System safety subpanel

FIGURE 12.9
NASA relies on a multilayered system of panels and boards for decisions on engineering design and safety matters. (Source: Reprinted with permission from *Post-Challenger Evaluation of Space Shuttle Risk Assessment Management*, 1988. Courtesy of the National Academy Press. Washington, D.C.)

- Organizational Roles
  - In theory, safety is equally the responsibility of all NASA managers, workers, and contractors.
  - In practice, roles and responsibilities are necessarily defined and allocated across various functional organizations:
    1. The engineering organizations within the project offices
    2. A Safety, Reliability, Maintainability, and Quality Assurance organization at headquarters and corresponding to the Safety, Reliability, and Quality Assurance organizations at each of the centers
    3. The Engineering Integration Office
    4. The operations organizations

- Safety-Related Analyses
  - FMEA and CIL (Critical Items List) are at the heart of NASA's effort to ensure reliability and safety.

- FMEA
  - Reliability analyses
  - Performed on all STS hardware, but not performed on software.
  - All identified items are categorized as Table 12.2

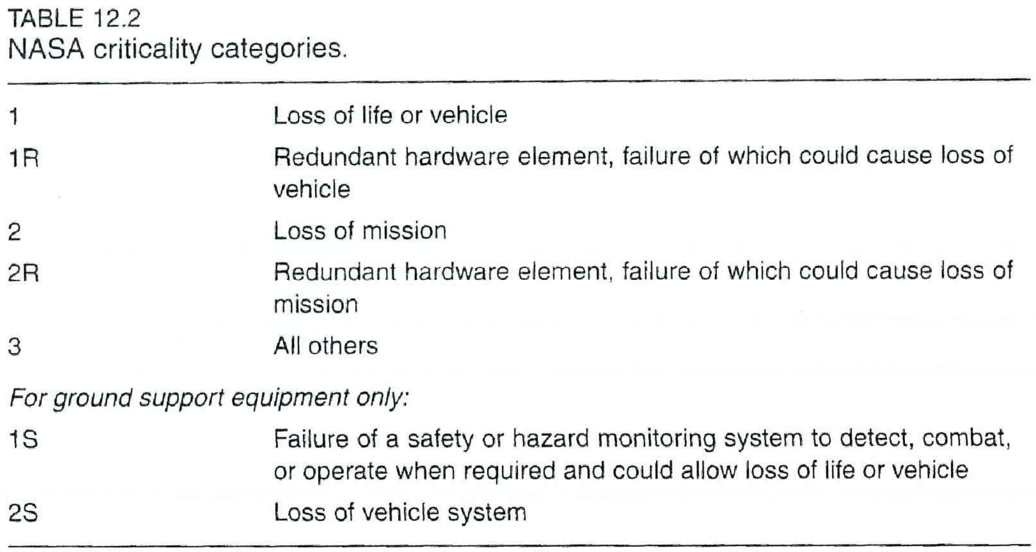
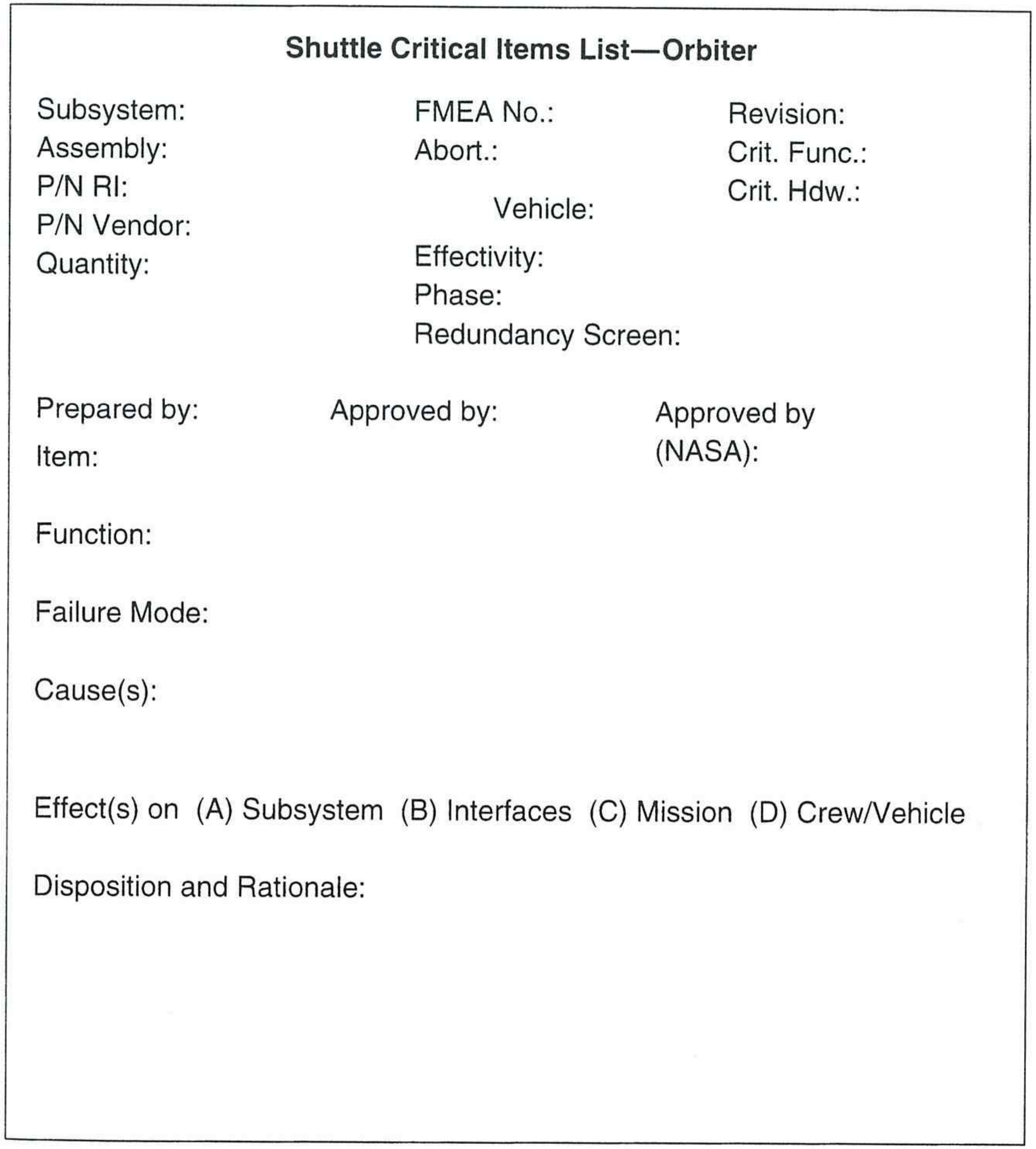


TABLE 12.2
NASA criticality categories.

1	Loss of life or vehicle
1R	Redundant hardware element, failure of which could cause loss of vehicle
2	Loss of mission
2R	Redundant hardware element, failure of which could cause loss of mission
3	All others
For ground support equipment only:	
1S	Failure of a safety or hazard monitoring system to detect, combat, or operate when required and could allow loss of life or vehicle
2S	Loss of vehicle system



Shuttle Critical Items List—Orbiter

Subsystem:
Assembly:
P/N RI:
P/N Vendor:
Quantity:

FMEA No.:
Abort.:
Vehicle:
Effectivity:
Phase:
Redundancy Screen:

Revision:
Crit. Func.:
Crit. Hdw.:

Prepared by:
Approved by:
Approved by (NASA):

Item:

Function:

Failure Mode:

Cause(s):

Effect(s) on (A) Subsystem (B) Interfaces (C) Mission (D) Crew/Vehicle

Disposition and Rationale:

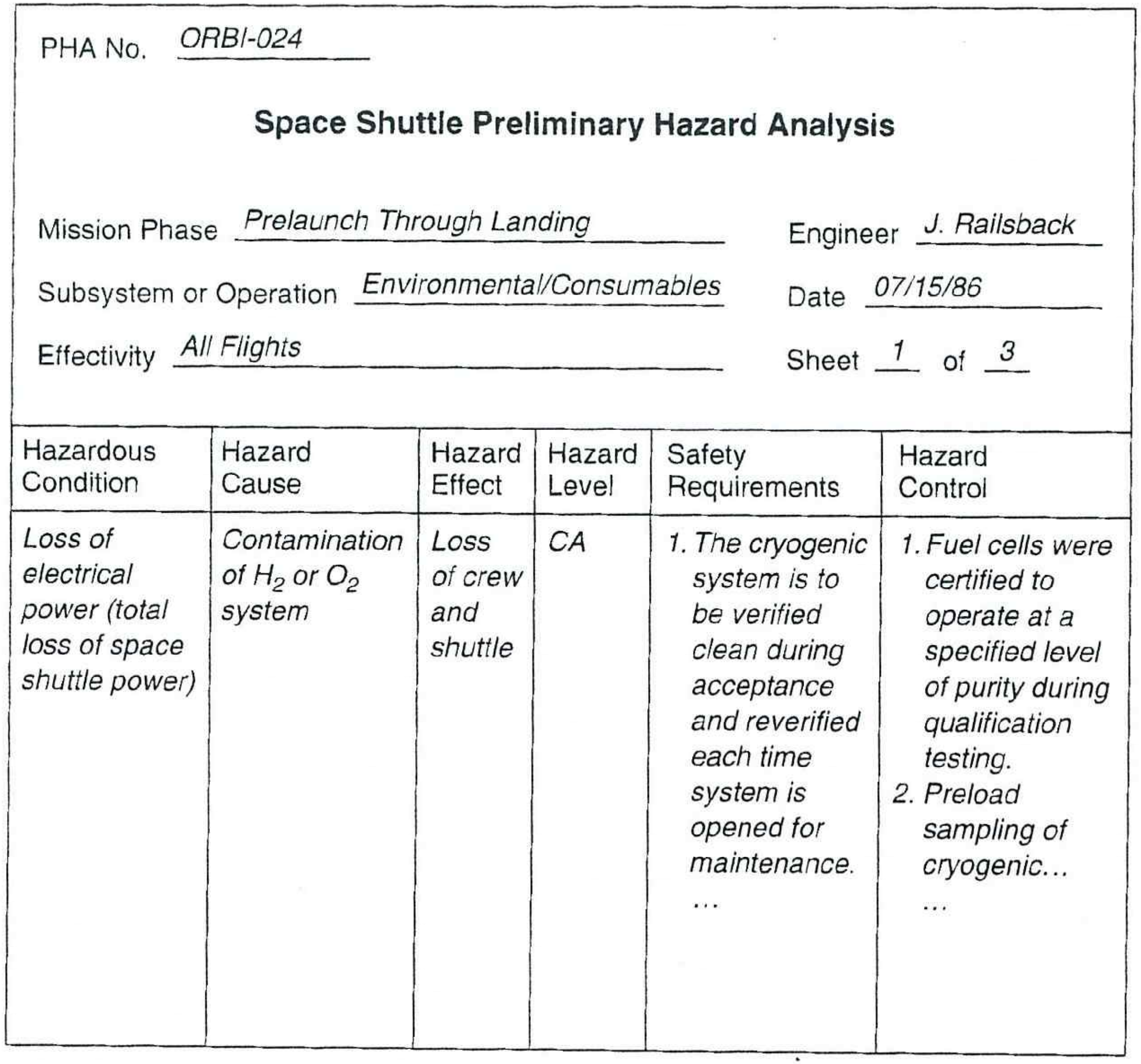
FIGURE 12.10
A Critical Items List form.

- FMEA
  - Items on the CIL must have design improvements or corrective action.
  - Element Interface Functional Analysis
    - Used to examine functional failure modes at the interface of two components.

- Hazard analyses
  - Used to consider
    - not only the failures identified in the FMEA
    - but other potential threats posed by the environments, crew-machine interfaces, and mission activities.

    - Standard Preliminary Hazard Analysis (Figure 12.11)
    - System Hazard Analysis
    - Subsystem Hazard Analysis
    - Operational Hazard Analysis

  - Software hazard analysis is not performed.



PHA No. *ORBI-024*

Space Shuttle Preliminary Hazard Analysis

Mission Phase *Prelaunch Through Landing*

Engineer *J. Railsback*

Subsystem or Operation *Environmental/Consumables*

Date *07/15/86*

Effectivity *All Flights*

Sheet *1* of *3*

Hazardous Condition	Hazard Cause	Hazard Effect	Hazard Level	Safety Requirements	Hazard Control
Loss of electrical power (total loss of space shuttle power)	*Contamination of H_2 or O_2 system*	*Loss of crew and shuttle*	*CA*	*1. The cryogenic system is to be verified clean during acceptance and reverified each time system is opened for maintenance.* *...*	*1. Fuel cells were certified to operate at a specified level of purity during qualification testing.* *2. Preload sampling of cryogenic...* *...*

FIGURE 12.11

Excerpt from a sample Space Shuttle Preliminary Hazard Analysis report.(Source: Reprinted with permission from *Post-Challenger Evaluation of Space Shuttle Risk Assessment Management*, 1988. Courtesy of the National Academy Press. Washington, D.C.)

- National Research Council report [53]
  - Criticized the disconnect between system safety program and the software development and maintenance activities.

- Data Collection and Reports
  - NASA used a number of special reports and reporting systems.
  - Problem Reporting and Corrective Action (PRACA) system
    - A large distributed database
  - A central database
    - To integrate a number of existing information systems to provide an integrated view.

- Post-Challenger Changes and Recommendations
  - Software still seems to be ignored.
  - [247] in 1988, focused on Shuttle risk management.
  - [53] in 1993, focused on the Shuttle software process.

Chapter 13

HAZARD ANALYSIS

Introduction

The argument that the same risk was flown before without failure is often accepted as an argument for the safety of accepting it again. Because of this, obvious weaknesses are accepted again and again, sometimes without a sufficiently serious attempt to remedy them, or to delay a flight because of their continued presence.

- Richard Feynman
Personal Observations on Reliability of Shuttle

Unfortunately, everyone had forgotten why the branch came off the top of the main and nobody realized that this was important.

-Trevor Kletz
-What Went Wrong?

- Hazard analysis is at the heart of any effective safety program, providing visibility and coordination.
  - A necessary first step before hazards can be eliminated or controlled through design or operational procedures

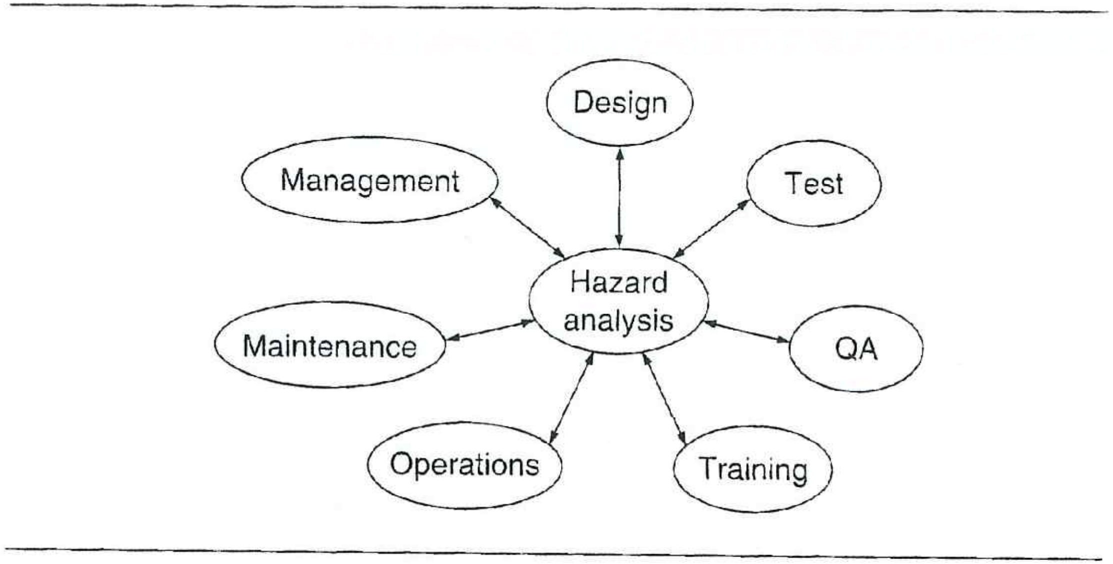


FIGURE 13.1
Hazard analysis provides visibility and coordination.

- Simply knowing that a hazard exists may provide sufficient information to eliminate or control it.
  - If more is known about the hazard, conditions and events leading to it,
    - A larger number of options for elimination and control exist.
      - Conditions-specific safeguards
      - Event-specific safeguards

- Hazard analysis
  - Not just performed at the beginning of a project or at fixed stages
  - Should be continuous throughout the life of the system
    - With increasing depth and extent as more information is obtained.

- In some countries or industries,
  - Particular hazards to be analyzed and the depth and the types of analysis may be established
  - By regulatory authorities or legislation

- In others,
  - These decisions must be made by the safety engineers early in project development.
  - This chapter presents the information used in making such decisions.

13.1 The Hazard Analysis Process

- Different goals of hazard analysis will require very different processes.
  - Once the goals are defined, the steps to be taken can be determined.

13.1.1 Goals of Hazard Analysis

- The goals of safety analysis
  - Making the system safer
  - Convincing management or government that existing design or system is safe

- These are related to three general tasks [323, 327]
  1. Development
     - The examination of an existing system to identify hazards and eliminate or control them.
     - "Making the system safer"
  2. Operational management
     - The examination of an existing system to identify and assess hazards in order to improve the level of safety, to formulate a safety management policy, to train personnel, and to increase motivation for efficiency and safety of operation
     - "Making the system safer"
  3. Certification
     - The examination of a planned or existing system to demonstrate its level of safety and to be accepted by the authorities or the public.
     - "Convincing management or government that existing design or system is safe"

- These three broad tasks can be divided into subtasks:
  - Development and operational management
    - 1 ~ 11
  - Certification
    - 1 ~ 2
  - Refer to the text (pp.289~290)

- Each of these subtasks implies very different types of analysis.
  - The goals and subtasks need to be clarified and documented at the start of any hazard analysis process.

13.1.2 Qualitative versus Quantitative Analyses

- Quality of quantitative analysis depends on how good the qualitative one was.
  - Even if quantitative methods are used, qualitative analyses must precede them.
  - Hazards and their causal factors must be identified before numerical values can be assigned to them.
  - Analysis can stop with the qualitative aspects.
    - When few accurate numerical values are available anyway.

- The use of probabilistic risk assessment has been debated widely.
  - Based on the technique's ability to provide input for decision making.

  - Many important factors (design deficiencies) cannot be easily or reasonably quantified.
    - Equipment failures are the most easily assessed probabilistically.
    - Many quantitative risk assessments have been criticized for placing more emphasis on these failures than on less easily predicted and quantified factors
      - Design errors, construction deficiencies
      - Operator actions, maintenance errors, management deficiencies

  - System safety managers may become so enamored with the statistics that simpler and more meaningful engineering processes are ignored.

- The emphasis of this chapter and this book, is on qualitative rather than quantitative hazard analysis.

13.1.3 Role and Qualifications of the Analyst

- The role of the safety analysis should be to generate alternatives as well as to eliminate them.
  - Pointing out new alternatives may be more valuable to a program than exhaustive analysis of given approaches [5].

- The job of the system safety analyst is
  - to dig deeply into the design and
  - Suggest designs that will yield eventual system operation that is satisfactory from both a safety and performance standpoint.

13.1.4 General Features of an Effective Hazard Analysis Process

- The hazard analysis process is both continual and iterative.

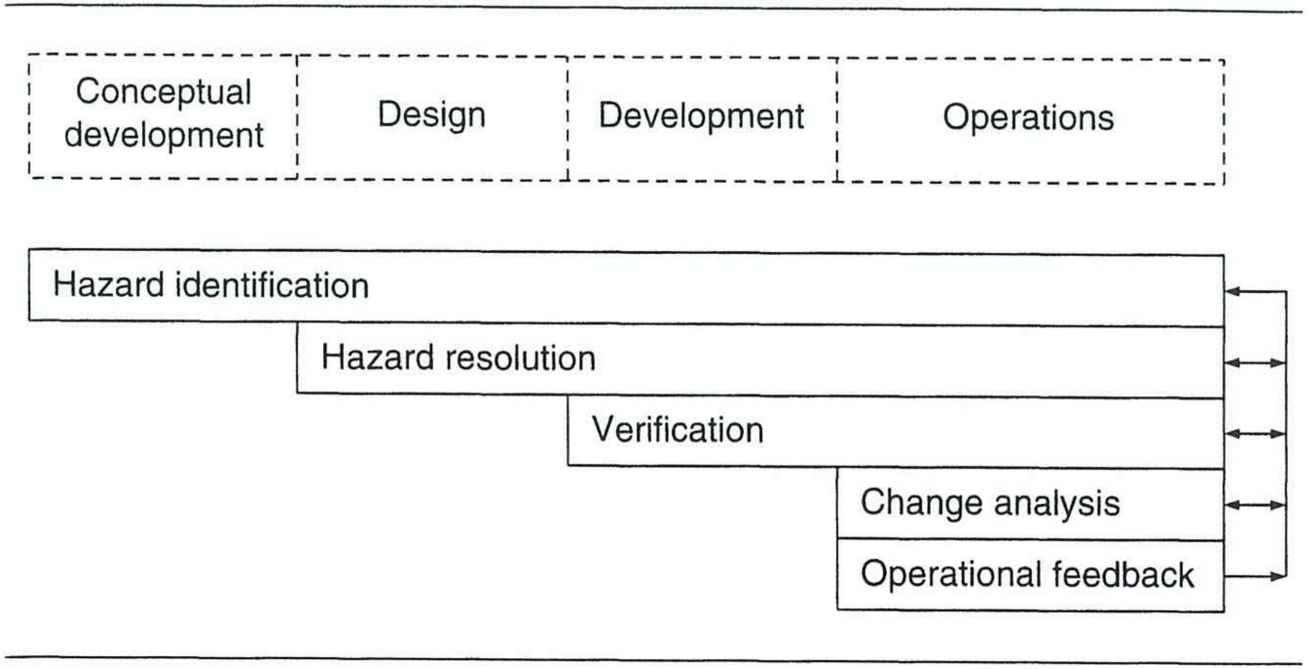


FIGURE 13.2
The hazard analysis process is continual and iterative.

- The forms of analysis will be different as the system matures, but all are part of a single analysis process.

- If a hazard cannot be resolved at a particular stage, follow-up evaluation and review may be necessary.

- The operational safety achieved depends on the accuracy of the assumptions and models underlying the design process.
  - The system must be monitored to ensure
    - (1) It is constructed, operated, and maintained in the manner assumed by the designers.
    - (2) The models and assumptions used during initial decision making and design were correct.
    - (3) The models and assumptions are not violated by changes in the system, such as workarounds or unauthorized changes in the procedures, or by changes in its environments.

- If a change is proposed to a baseline or completed design
  - It must be analyzed for its potential effect on safety.
    - Reviewing previously generated analyses to identify the impact of the changes
    - Updating and documentation to reflect the changes
    - Identifying new hazards and hazard causes

13.1.5 Steps in the Process

- A hazard analysis consists of the following steps:
  1. Definition of objectives
  2. Definition of scope
  3. Definition and description of the system, system boundaries, and information to be used in the analysis
  4. Identification of hazards
  5. Collection of data (such as historical data, related standards and codes of practice, scientific tests and experimental results)
  6. Qualitative ranking of hazards based on their potential effects (immediate or protracted) and perhaps their likelihood (qualitative or quantitative)
  7. Identification of causal factors
  8. Identification of preventive or corrective measures and general design criteria and controls
  9. Evaluation of preventive or corrective measures, including estimates of cost
  10. Verification that controls have been implemented correctly and are effective
  11. Quantification of selected, unresolved hazards, including probability of occurrence, economic impact, potential losses, and costs of preventive or corrective measures
  12. Quantification of residual risk
  13. Feedback and evaluation of operational experience

- Each step requires documentation of the results and of any underlying assumptions and models.

- The purpose of the hazard analysis process is
  - To use the results as a reference for judgment [274] in design, maintenance, and management decisions.
  - Requires an explicit formulation of the models, premises, and assumptions underlying the safety analysis and the design features used to eliminate or control hazards.

- Not all of the steps may need to be performed for every system and for every hazard.

- Hazard analysis can be divided into three basic functions:
  (1) Identifying hazards
  (2) Identifying and evaluating the hazard causal factors
  (3) Evaluating risk

13.1.6 Hazard Identification

- Hazard identification starts early in the concept formation stages.
- Hazard lists are continually updated throughout the entire lifetime of the system.

- PHA (Preliminary Hazard Analysis)
  - Hazard identification in the earliest stage of a program
  - Involves
    1. Determining what hazards might exist during operation of the system and their relative magnitude.
    2. Developing guidelines, specifications, and criteria to be followed in system design.
    3. Initiating actions for the control of particular hazards.
    4. Identifying management and technical responsibilities for action and risk acceptance and assuring that effective control is exercised over the hazards.
    5. Determining the magnitude and complexity of the safety problems in the program.
  - The results serve as
    - A framework or baseline for later analyses
    - A checklist to ensure that management and technical responsibilities for safety tasks are carried out

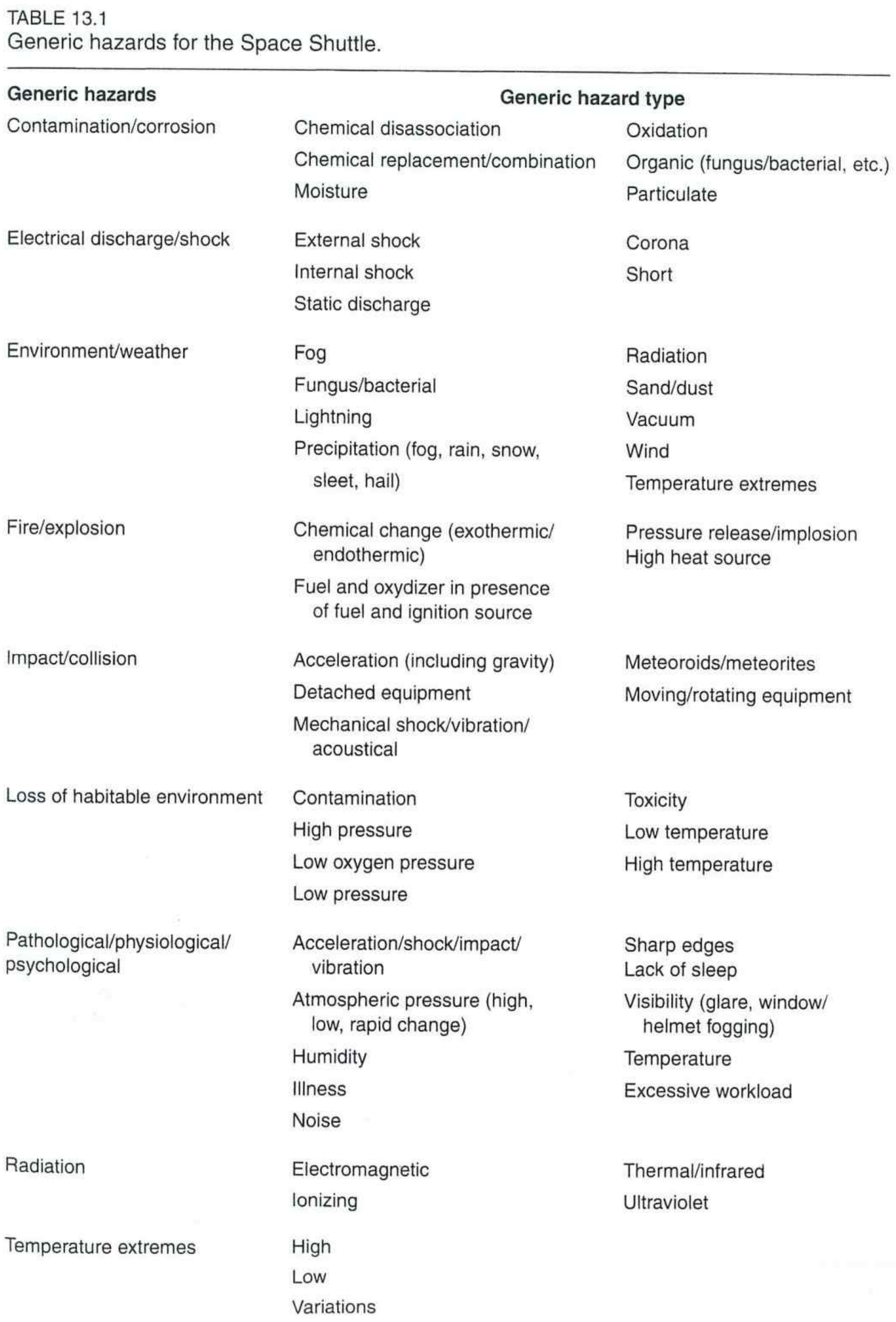


TABLE 13.1
Generic hazards for the Space Shuttle.

Generic hazards	Generic hazard type	
Contamination/corrosion	Chemical disassociation	Oxidation
	Chemical replacement/combination	Organic (fungus/bacterial, etc.)
	Moisture	Particulate
Electrical discharge/shock	External shock	Corona
	Internal shock	Short
	Static discharge	
Environment/weather	Fog	Radiation
	Fungus/bacterial	Sand/dust
	Lightning	Vacuum
	Precipitation (fog, rain, snow, sleet, hail)	Wind
		Temperature extremes
Fire/explosion	Chemical change (exothermic/ endothermic)	Pressure release/implosion High heat source
	Fuel and oxydizer in presence of fuel and ignition source	
Impact/collision	Acceleration (including gravity)	Meteoroids/meteorites
	Detached equipment	Moving/rotating equipment
	Mechanical shock/vibration/ acoustical	
Loss of habitable environment	Contamination	Toxicity
	High pressure	Low temperature
	Low oxygen pressure	High temperature
	Low pressure	
Pathological/physiological/ psychological	Acceleration/shock/impact/ vibration	Sharp edges Lack of sleep
	Atmospheric pressure (high, low, rapid change)	Visibility (glare, window/ helmet fogging)
	Humidity	Temperature
	Illness	Excessive workload
	Noise	
Radiation	Electromagnetic	Thermal/infrared
	Ionizing	Ultraviolet
Temperature extremes	High	
	Low	
	Variations	

- For some special systems regulated by government agencies, hazards or hazard categories may be mandated.
  - The U.S. DoD (for hazards in Section 12.2.2)
  - NSCCA (Nuclear Safety Cross Check Analysis)
  - SNSA (Software Nuclear Safety Analysis)

- However, in most systems, the hazards are not immediately known or subject to government mandate and need to be determined.

- A few techniques for identifying hazards have been developed.
  - Structured ways to stimulate a group of people to apply their personal knowledge to the task.
  - Most hazard identification involves less structured procedures
    - Review pertinent historical safety experience, lessons learned, etc.
    - Use published lists and checklists of hazards (Table 13.1)
    - Examine the basic energy sources, energy flows, …
    - …
    - Think through the entire process, step by step, …
    - (See page 296)

- As hazards are identified, information about them needs to be recorded.
  - Tabular forms are often used.
  - To use one form, filling it in as the analysis progresses through the various stages.

- The form may include some or all of the following information:
  - System, subsystem, unit
  - Hazard description
  - Hazard cause
  - Possible effects on the system and the environment
  - Category (hazard level)
  - Corrective or preventive measures, possible safeguards, recommended action, and design criteria
  - Operational phase when hazardous
  - Organizations responsible for ensuring that safeguards are provided for the specific hazard
  - Verification methods to verify that the hazard is effectively controlled
  - Other proposed and necessary actions
  - Remarks and status of the hazard resolution process

- Hazard Level
  - The hazard category or level is defined by *likelihood* and *severity*.

- Hazard severity categories reflect worst possible consequences.
  - Specific to the industry and sometimes the system
  - The DoD (MIL-STD-882B: System Safety Program Requirements)
    - Category I : Catastrophic
    - Category II : Critical
    - Category III : Marginal
    - Category IV : Negligible
  - The NASA (NHB 5300.4)
    - Category 1 : Loss of life or vehicle
    - Category 2 : Loss of mission
    - Category 3 : All others
  - The Department of Energy standard (DOE 548.1)
    - High : Hazards with potential for major onsite or offsite impacts to people or the environment
    - Moderate : At most only minor offsite impacts
    - Low : Minor onsite and negligible offsite impacts

- The likelihood is commonly divided into discrete categories, such as
  - Frequent : Continuously experienced
  - Probable : Frequently
  - Occasional : Several times
  - Remote : unlikely but reasonably expected to occur
  - Improbable : possible
  - Physically Impossible : Cannot occur to an item or in a fleet or inventory
  - Quantitative probability assessment, if used, such as 10^{-7} per year.
- Design Criteria
  - Broad concept states what has to be achieved.
  - Leaving the designer free to use ingenuity in deciding how the goal may best be achieved.
  - Table 13.2
  - Not requirements, but rather are used to derive the design requirements.

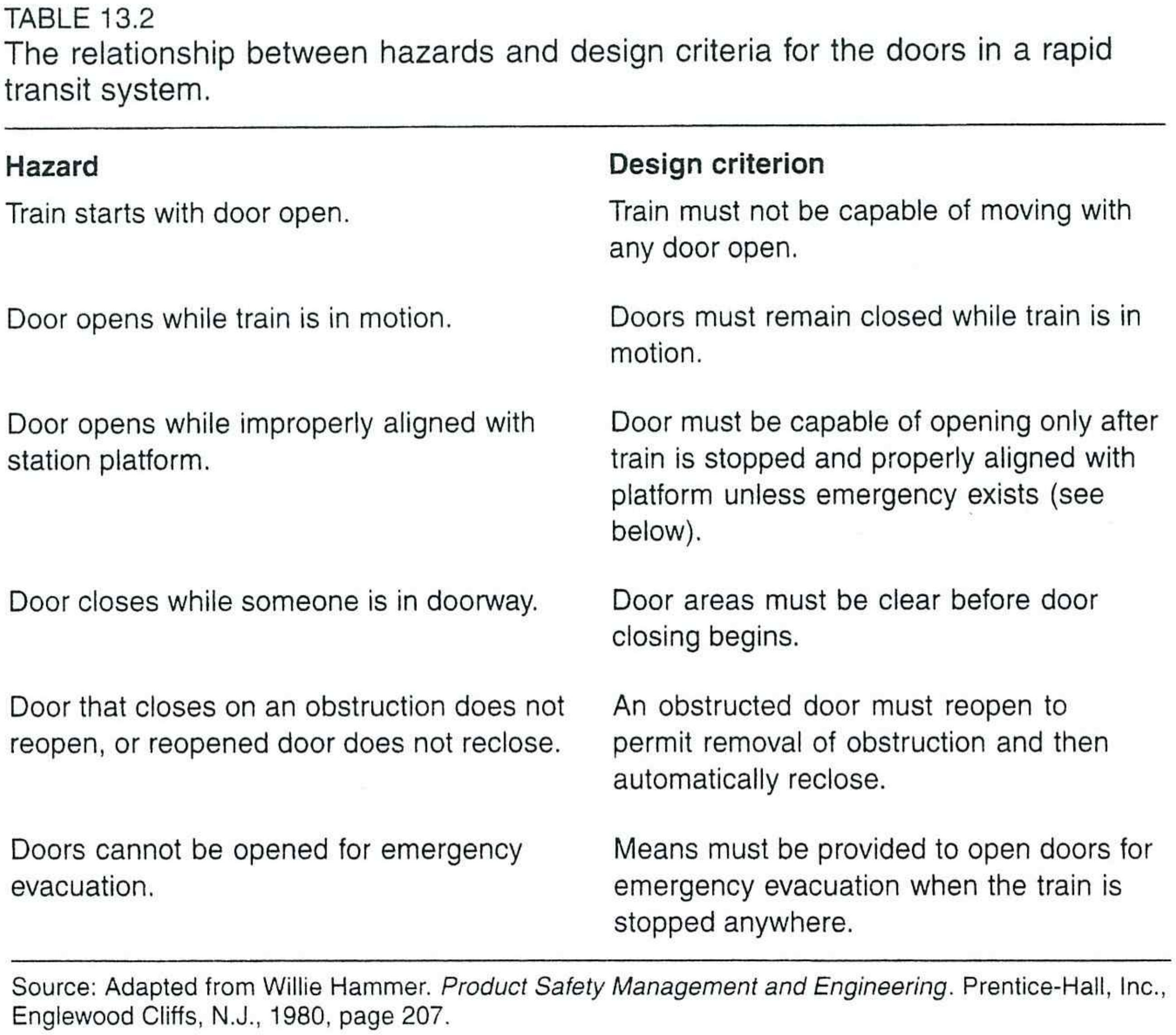


TABLE 13.2
The relationship between hazards and design criteria for the doors in a rapid transit system.

Hazard	Design criterion
Train starts with door open.	Train must not be capable of moving with any door open.
Door opens while train is in motion.	Doors must remain closed while train is in motion.
Door opens while improperly aligned with station platform.	Door must be capable of opening only after train is stopped and properly aligned with platform unless emergency exists (see below).
Door closes while someone is in doorway.	Door areas must be clear before door closing begins.
Door that closes on an obstruction does not reopen, or reopened door does not reclose.	An obstructed door must reopen to permit removal of obstruction and then automatically reclose.
Doors cannot be opened for emergency evacuation.	Means must be provided to open doors for emergency evacuation when the train is stopped anywhere.

Source: Adapted from Willie Hammer. *Product Safety Management and Engineering*. Prentice-Hall, Inc., Englewood Cliffs, N.J., 1980, page 207.

- Operational Phase
  - Whether an accident actually occurs as a result of a hazard and the severity of the hazard may depend on the operational phase in which the hazard occurs.
  - The system and environmental conditions
  - The phase must also be documented for each hazard.
  - Example:
    - A missile launch system

13.1.7 Hazard Causal Analysis

- The next step after hazards are identified
  - To determine the causes and effects associated with each hazardous condition
  - Helpful in specifying the safety requirements and design constraints needed to minimize or control hazards to an acceptable level.
  - The factors considered and the process of identifying them will depend on the underlying accident model used. (see Chapter 10)
  - System Hazard Analysis
  - Subsystem Hazard Analysis
  - Software Hazard Analysis
  - Operational Hazard Analysis

- System Hazard Analysis
  - Consider the system as a whole
  - Identify how the system operation, the interfaces between the system components, and the interfaces between the system and its operators can contribute to hazards.

- Subsystem Hazard Analysis
  - Look at individual subsystems and determines the effect of their operating or failure modes on system hazards.
  - Includes power, structural, control, sensor, operator, communications, propulsion and environmental control

- Software Hazard Analysis
  - Software is just like any other component, especially various types of control or monitoring components.
  - Will be included in SHA or SuHA.
  - Later, the software is evaluated to determine.
    - If the system design criteria have been satisfied in the software specifications
    - If the software implementation has impaired or degraded system safety or introduced new hazards

- Operational Hazard Analysis
  - A separate analysis of the risk controls that have been assigned to operational procedures.
  - Often it is left until the design is completed.
  - However, not different than the other analyses.
  - The operational procedures are defined in concert with the automated systems.
    - Not consist merely of functions that the designers did not know how to automate.
  - It looks at all the ways that the operators can contribute to system hazards if they follow defined procedures or if they do not.
    - Typically, operational procedures are divided into phases or tasks.

13.1.8 Risk Assessment and Acceptance Analysis

- The goal is to evaluate the final product.
  - The result may be used internally or for independent certification to determine the residual risk and whether the system is acceptable for use.

- Risk assessment is a description of the potential problem and what is being done about it.
  - The document should allow the assessor or reviewer to follow the analyst's reasoning in order to check the correctness of the results. [299]

- Acceptance analysis also includes a probabilistic risk assessment.
  - The probability and severity of accidents are evaluated in addition to the hazard level.
  - However, requires elaborate quantitative analyses.
    - Unfortunately, obtaining probabilistic data about the harmful consequences of some hazards and about human error is quite difficult.
      - Assessing harmful consequences
      - Assessing human error

13.2 Types of System Models

- Every hazard analysis requires some type of model of the system.
  - May range from a high-level abstraction to a low-level and detailed prototype.

- A model
  - A representation of a system that can be manipulated in order to obtain information about the system itself.
  - Categorized along different dimensions [50]:
    - Material models vs. Symbolic or Formal models
    - Dynamic models vs. Static models
    - Stochastic models vs. Deterministic models
    - Iconic models vs. Analog models vs. Symbolic models
  - Modeling any system requires a description of the following:
    - Structure
    - Distinguishing qualities
    - Magnitude, probability, and time
  - This information can be stored or displayed in various ways.
    - Simple tables, bubble charts, trees, graphs, state machines, algebraic equations, etc.

- Selection of the model will determine
  - What information can be specified.
  - What can be derived from the model through analysis.

- It is important to specify
  - The boundaries of the system being modeled and what is not included
  - Assumptions about the system
  - What assumptions are most likely to be incorrect or invalidated by changes, and thus need to be checked periodically.

13.3 General Types of Analysis

- Different types of models allow for various types of analysis or manipulation of the model to learn more about a system.
  - Tradeoff between the difficulty of building and analyzing the model and the quality of information that can be derived from it
  - No one model or analysis technique is useful for all purpose.
    - More than one type may be required on a project.

- Analysis techniques can be differentiated by
  - goals
  - whether they are quantitative
  - phase in the life cycle when they are used
  - depth of analysis
  - domain upon which they are defined
  - search methods used

- Analysis techniques usually involve searching
  - Forward and backward
  - Top-down and bottom-up
  - Combinations of these two

13.3.1 Forward and Backward Searches

- Forward (inductive) and backward (deductive) searches are useful, when
  - The underlying structure is temporal, and
  - The elements are events, conditions, or tasks.

- Forward search
  - Takes an initiating event (or condition) and traces it forward in time.
  - The result is a set of states that represent the effects of the initiating event.
  - To look at the effect on the system state of both
    (1) An initiating event
    (2) Later events that are not necessarily caused by the initiating event.

  - Tracing an event forward can generate a large number of states.
    - Often limited to only a small set of temporally ordered events.

- Backward search
  - Starts with a final event or state and determines the preceding events or states.
  - Fits well with chain-of-event accident models
    - The goal is to determine the paths that can lead to a particular hazard or accident.
  - Useful in accident investigations and in eliminating hazards by installing controls to eliminate predecessor events.

- The results of forward and backward searches are different.
  - Backward search
    - If the goal is to explore the precursors of a specific hazard or accident,
    - One particular hazard or accident
    - Multiple initiating events
  - Forward search
    - If the goal is to determine the effects of a specific failure,
    - One initiating state
    - Multiple final states

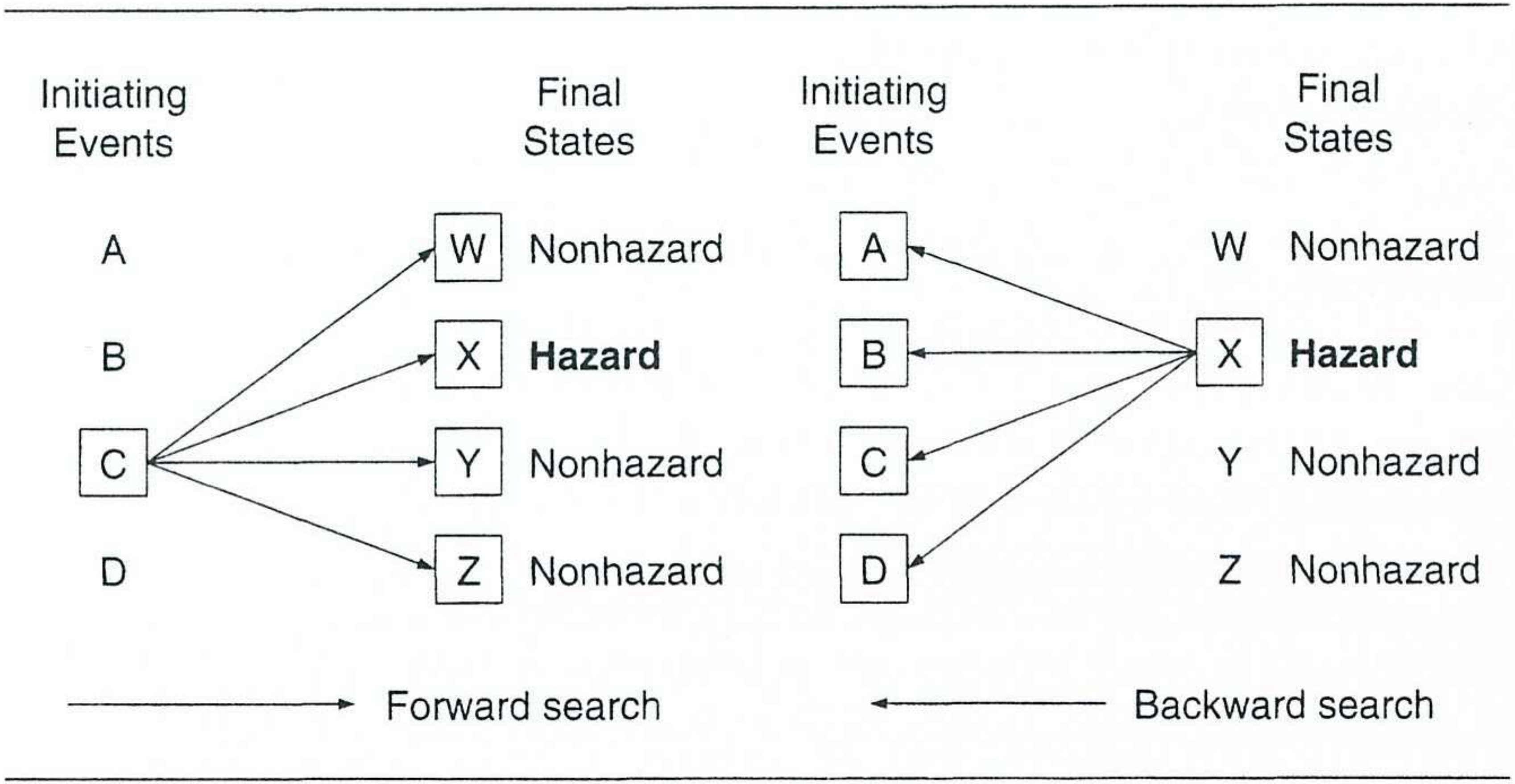


FIGURE 13.3
The states found in a forward search and in a backward search will probably not be the same.

13.3.2 Top-Down and Bottom-Up Searches

- Relationship being investigated here is structural (whole-part)
  - Higher-level abstractions are refined or broken down into their constituent parts.
  - Top-down search
    - A basic event, service, task, or system may be broken up into more basic events, conditions, tasks, or subsystems.
    - Example: Identification of all the ways that power can be lost
  - Bottom-up search
    - Subcomponents are put together in different ways to determine the result.
    - Example: Examine the effect of an individual battery failures on the system as a whole
- The results of top-down and bottom-up searches are not the same.

13.3.3 Combined Searches

- Starts with some event or deviation
  - And goes forward and backward or top-down and bottom-up to find paths between hazards (or accidents) and their causes or effects.

13.4 Limitations and Criticisms of Hazard Analysis

- Understanding hazard analysis's limitations and common problems is important.
  - Since hazard analysis serves as the basis of judgment for many aspects of system safety.
  - Qualitative analysis always precedes quantitative analysis.
    - All limitations of the former apply to the latter.
    - The latter has additional limitations, however.

- Three limitations

  (1) Hazard analysis may not match reality.

  (2) The modeling techniques are related to oversimplification.

  (3) Must be used by humans.

- Hazard analysis, along with their underlying accident and system models, may not match reality.
  - Often make unrealistic assumptions.
  - Even if all the assumptions are right to begin with, conditions change and the models may not accurately reflect the current system.
    - In general, there is no way to assure completeness.
    - All factors cannot be considered.
      - [113], in practice, the majority of errors in hazard analyses result from faults in the model or failures to foresee hazards and not from errors in data.

- Simplifications in the modeling techniques
  - Policy and principles of management are rarely included.
  - Many important factors requiring of assigning numbers are omitted.
  - Some stem from limitations in knowledge.
  - Etc.

- Limitations stemmed from the fact that they must be used by humans.

- With this information about the hazard analysis process and its general limitations in mind,
  - We are ready to examine specific hazard analysis models and techniques.

Chapter 14

HAZARD ANALYSIS MODELS AND TECHNIQUES

Introduction

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 Mushle

- Many different types of hazard analysis have been proposed and used.
  - Selecting appropriate models and techniques that match the project’s goal, tasks, and skills is important.
  - Very little validation of these techniques has been done.

  - They must be used carefully and combined with a large dose of engineering judgment and expertise.

14.1 Checklists

- Repository of mistakes made and lessens learned
  - Checklists are one way to pass on hard-earned experience in engineering so that each project need not relearn the lessons of the past and start each hazard analysis from the scratch.
  - Checklists provide feedback to engineering process.

  - May also be derived from standards and codes of good engineering practices.
  - Most useful in the design of well-understood systems,
    - Which standard design features and knowledge have been developed over time.

- Commonly used in all life-cycle phases.

- Checklists are an excellent way to pass on lessons learned, especially for hazard identification.
  - For designers
    - Help to ensure good engineering design practices and compliance with standards, codes, and specifications.
  - For reviewers
    - 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.
  - Checklists can become large and difficult to use.
  - Checklists often include false confidence - a belief that if everything is checked off, the system is safe.
  - Problems may arise when the lists are used without giving careful thought to the specific situation being considered.

14.2 Hazard Indices

- Hazard indices measure 'loss potential' due to fire, explosion, and chemical reactivity hazards in the process industries.
  - Developed primarily for insurance purposes, but can be useful in
    - General hazard identification,
    - Assessing hazard level for certain well-understood hazards,
    - Selecting hazard reduction design features for the hazards reflected in the index,
    - Auditing an existing plant.
  - Dow Index
    - The oldest and most widely used index, in 1964.
    - The *Dow Chemical Company Fire and Explosion Index Hazard Classification Guide*

- Hazard indices provides a quantitative indication of the potential for hazards associated with a given design.
  - Work well in the process industry
    - Where, designs and equipment are standard and change little
  - Less useful
    - for systems where designs are unique and technology change rapidly

- The hazard indices
  - Only consider a limited set of hazards.
  - 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.

14.3 Fault Tree Analysis

- Fault tree analysis is a top-down search method. (Figure 14.1)
  - Widely used in aerospace, electronics, and unclear industries.
  - A means for analyzing causes of hazards, not identifying hazards.
    - Top event : must have been foreseen and thus identified first by other techniques
    - Uses Boolean logic to describe the combinations of individual faults that can constitute a hazardous event.
    - 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 needed, the frequency of the top event may be calculated.

- FTA has four basic steps:

  (1) System definition

  (2) Fault tree construction

  (3) Qualitative analysis

  (4) Quantitative analysis

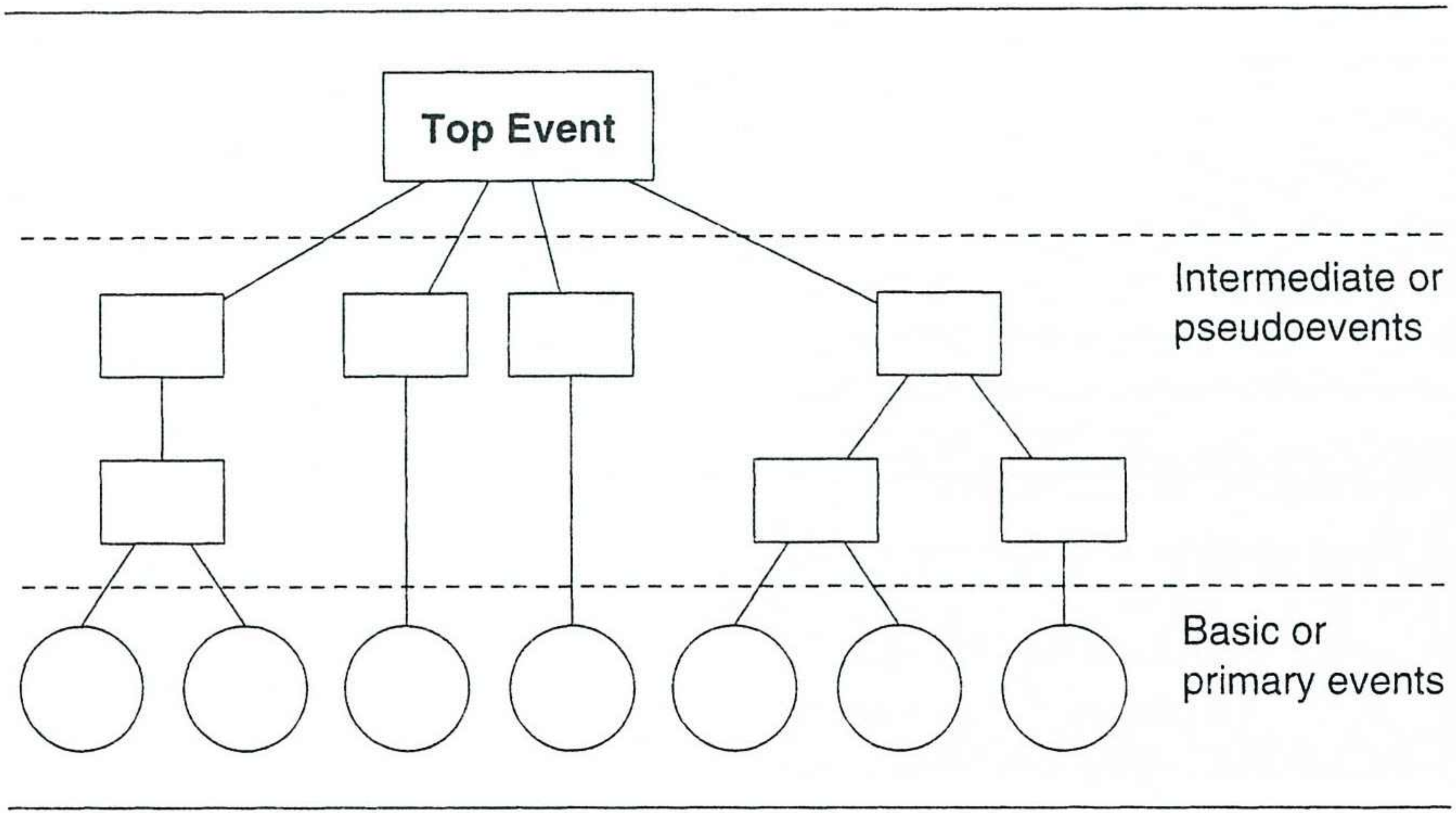
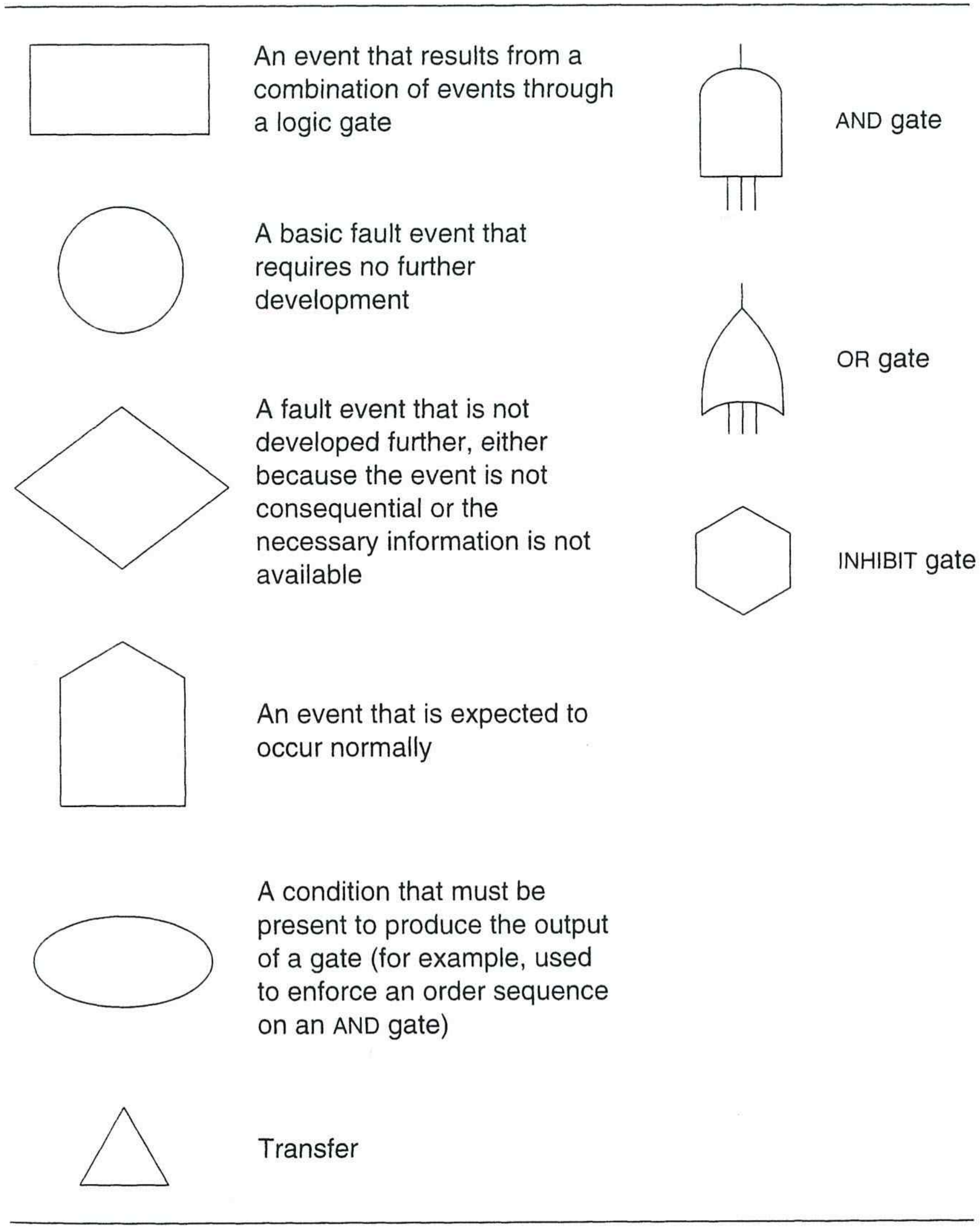


FIGURE 14.1
The leaf nodes of a fault tree represent the basic or primary events.

- System definition
  - The most difficult part of the FTA task
    - Requires determining the top event, initial conditions, existing events, and impermissible events.
    - The selection of top events is crucial.
  - A thorough understanding and definition of the system and its interrelationships is essential.
    - The analyst may use
      - System functional diagrams, flow diagrams, logic diagrams, other design representations,
      - His or her knowledge of the system
    - The physical system boundary must be carefully defined.
- Fault tree construction
  - 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)



An event that results from a combination of events through a logic gate

A basic fault event that requires no further development

A fault event that is not developed further, either because the event is not consequential or the necessary information is not available

An event that is expected to occur normally

A condition that must be present to produce the output of a gate (for example, used to enforce an order sequence on an AND gate)

Transfer

AND gate

OR gate

INHIBIT gate

FIGURE 14.2
Fault tree symbols.

- Fault tree construction
  - Continues, with each level of the tree considered in turn until basic or primary events are reached.
    - The analysts must determine the stopping rule for the analysis.
    - Stooping rule = resolution limit of the analysis
  - Figure 14.3

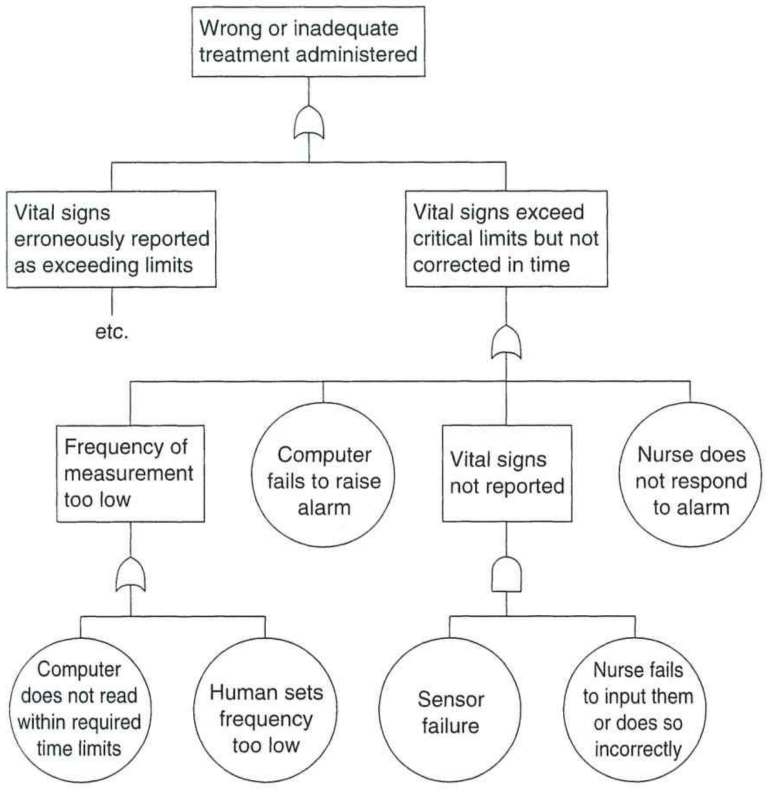

Wrong or inadequate treatment administered
Vital signs erroneously reported as exceeding limits
etc.
Vital signs exceed critical limits but not corrected in time
Frequency of measurement too low
Computer fails to raise alarm
Vital signs not reported
Nurse does not respond to alarm
Computer does not read within required time limits
Human sets frequency too low
Sensor failure
Nurse fails to input them or does so incorrectly

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

- Qualitative analysis
  - After the tree is constructed, qualitative analysis can begin.
  - Reducing the tree to a logically equivalent form showing the specific combinations of basic events sufficient to cause the top event
    - Intermediate pseudo events are removed.
    - Only relationship between the top event and the primary events are described.
  - Goal: to find the minimal cut sets
    - Basic events that will cause the top event and which cannot be reduced in number.
    - A cut set that does not contain another cut sets
    - Provide information that helps identify weaknesses in the system.
  - Cut set
    - If even one event in the cut set does not occur, the top event will not take place.
  - A medium-sixed fault tree can have millions of minimal cut sets.
    - Computer programs have been developed to calculate them.
    - The procedure for reducing the tree to a logically equivalent form are beyond the scope of this book.

- Quantitative analysis
  - The probability of the output of a logical gate is equal to the probability of the corresponding function of the input events.
  - 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,
    - Only 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, the calculation becomes complicated.
- Automatic Synthesis
  - Only for systems consisting purely of hardware elements
  - Taylor's technique [335]
    - Takes the components of the hardware model and describes them as transfer statements.
    - The same type of analysis can be done using state-machine models. (Section 14.12)

- Software FTA
  - Fault tree analysis can be applied to software (Chapter 18), but not easy.
  - Probabilistic analysis is not applicable.

- Life-cycle phase
  - 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].
  - 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, forcing system-level examination beyond the context of a single component or subsystem.
    - Problems are also found because the analysts has to think about the system in great detail during tree construction.
  - Fault tree can help the analyst identify scenarios leading to hazards.
    - Suggest possibilities for hazard elimination or control even before any analysis is performed on the tree.
  - Knowing the minimum cut sets for a particular fault tree can provide valuable insight into potential weak points of a complex system.

- Fault tree analysis has several limitations.
  - The most useful fault trees can be constructed only after the product has been designed.
  - However, a good safety program requires concentration on the early stages of the system life cycle.

- Fault tree analysis shows cause and effect relationships, but little more.
  - Additional analysis and information is usually required for an effective safety program.
  - Reliability analysts usually concentrate only on failure events in fault trees, whereas hazard analysis requires a broader scope.

- A fault tree, like any other model, is a simplified representation of a generally very complex process.
  - Its relative simplicity can de deceptive [172].
  - Simple AND and OR gates do not convey any notion of time ordering or time delay.
  - Transitions between states are not represented in fault trees.
    - Partial failures and multiple failures can cause difficulties [62].

- Common-cause failures cause problem and can lead to orders-of-magnitude errors in the calculated failure probability, in quantitative aspects [217].

14.4 Management Oversight and Risk Tree Analysis

- Management Oversight and Risk Tree analysis (MORT)
  - Developed in the 1970s for the U.S. Nuclear Regulatory Agency
  - Discussed as an accident model in Chapter 10, but it can also be used as an accident investigation or hazard analysis technique.
  - MORT is a standard fault tree augmented by an analysis of managerial functions, human behavior, and environmental factors.
- It uses an extensive checklist of 1,500 basic events or factors that facilitates finding those safety problems included in the list. (Figure 10.10)
- MORT considers factors related to the organization, information system, management practices, and principles and goals of the enterprise.
  - However, MORT is not used very often, because of its complexity.

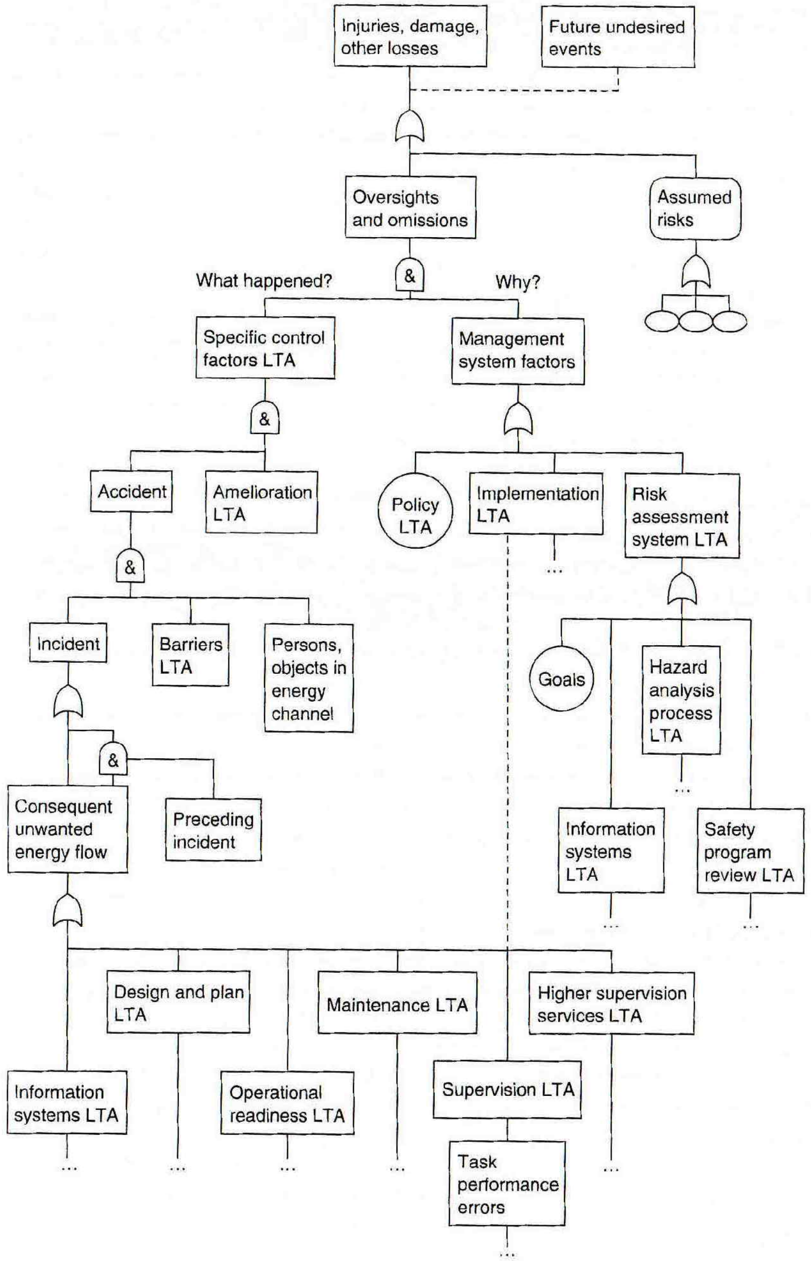


FIGURE 10.10
Top part of the MORT model. Each of the leaf nodes is expanded in further tree structures and each has a checklist associated with it. LTA stands for "less than adequate." (Source: Adapted from William G. Johnson. *MORT Safety Assurance*

14.5 Event Tree Analysis

- FTA is the most widely used method for the quantification of system failures, but it becomes very difficult to apply in complicated systems.

- Event Tree Analysis uses forward search to identify the various possible outcomes of a given initiating event, by determining all sequences of events that could follow it.
  - Drawn from left to right
  - With branches under each heading corresponding to two alternatives:
    1. Successful performance
    2. Failure
  - After the tree is drawn, path through it can be traced by choosing a branch under each successive heading, where each path corresponds to an accident sequence.

- Event trees tend to get quite large.

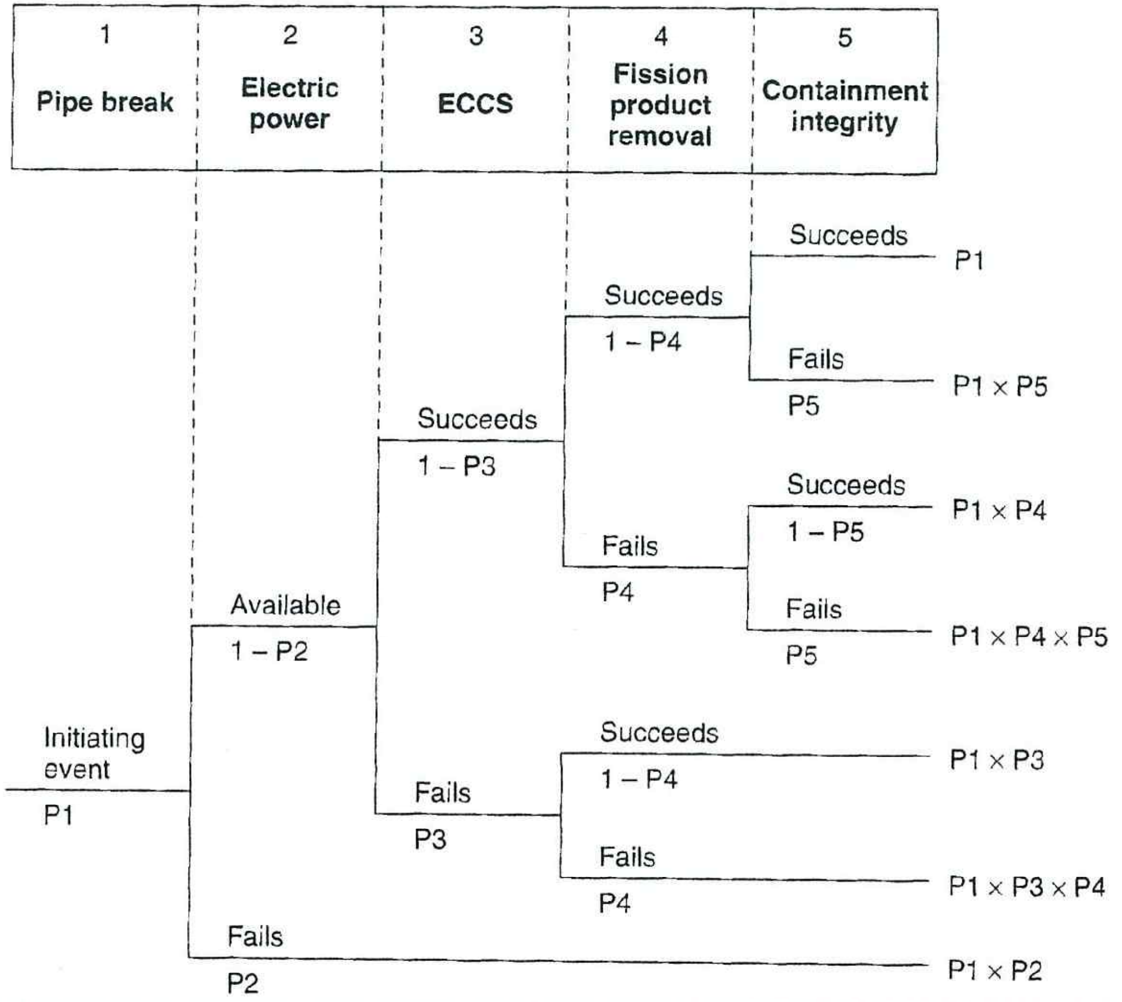


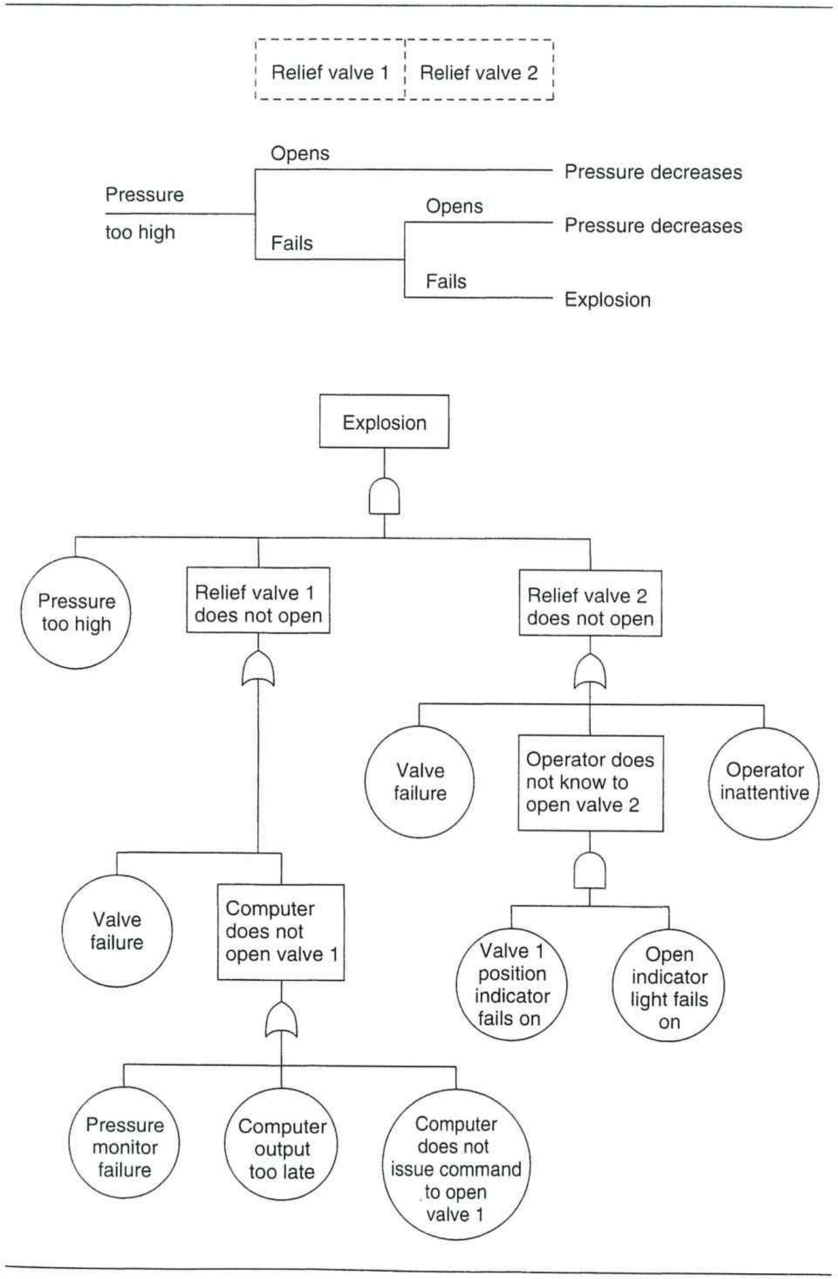
FIGURE 14.4
A reduced event tree for a loss of coolant accident. (Source: *Reactor Safety Study*, U.S. Nuclear Regulatory Commission, WASH-1400, NUREG 75/014, October 1974.)

- Event Tree Analysis is usually applied using a binary state system,
  - Where each branch of the tree has one failure state and one success state.
  - If a great number of discrete states are defined for each branch, then a branch must be defined for each state.
  - Timing issues can cause problems in event tree construction.
  - Possible dependencies between the various probabilities arising from common-cause failures should be considered.
- ETA is appropriate only after most of the design is complete.
  - Thus, it has been used primarily to evaluate existing plants or designs.
- Event trees
  - Display relationships between juxtaposed events (sequences of events) linked by conditional probabilities.
  - While, fault trees lay out relationships between events: They are snapshots of the system state.

- Event trees are, in theory, better at handling notions of continuity (logical, temporal, and physical).
  - While fault trees are more powerful in identifying and simplifying event scenarios.
  - In figure 14.5
    - A top-down search model like a fault tree loses the information about the ordering of relief valve operation,
    - 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.

- Limitations
  - Can become exceedingly complex.
  - Will require person-year of efforts
  - The use of FTA to determine the probability for many of the event tree branches may make it more difficult to identify common causes of failures.



Relief valve 1
Relief valve 2
Pressure too high
Opens
Pressure decreases
Fails
Opens
Pressure decreases
Fails
Explosion
Explosion
Pressure too high
Relief valve 1 does not open
Relief valve 2 does not open
Valve failure
Operator does not know to open valve 2
Operator inattentive
Valve failure
Computer does not open valve 1
Valve 1 position indicator fails on
Open indicator light fails on
Pressure monitor failure
Computer output too late
Computer does not issue command to open valve 1

FIGURE 14.5
A fault tree and event tree comparison.

- 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 effect of multiple initiating events.

- The usefulness of event trees depends on being able to define the set of initiating events that will produce all the important accident sequences.

- Continuous, non-action systems such as dams are not appropriate for event tree analysis, as with fault trees.

14.6 Cause-Consequence Analysis

- Cause-Consequence Analysis (CCA)
  - Combines several search modes [240].
  - Starts with a critical event and determine the causes of the event (using top-down and backward search) and consequences that could result from it (forward search).
  - Cause-consequence diagram shoes both time dependency and causal relationship among events. (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.

- Cause-consequence diagrams can be formalized to provide semi-automatic analysis methods [333].

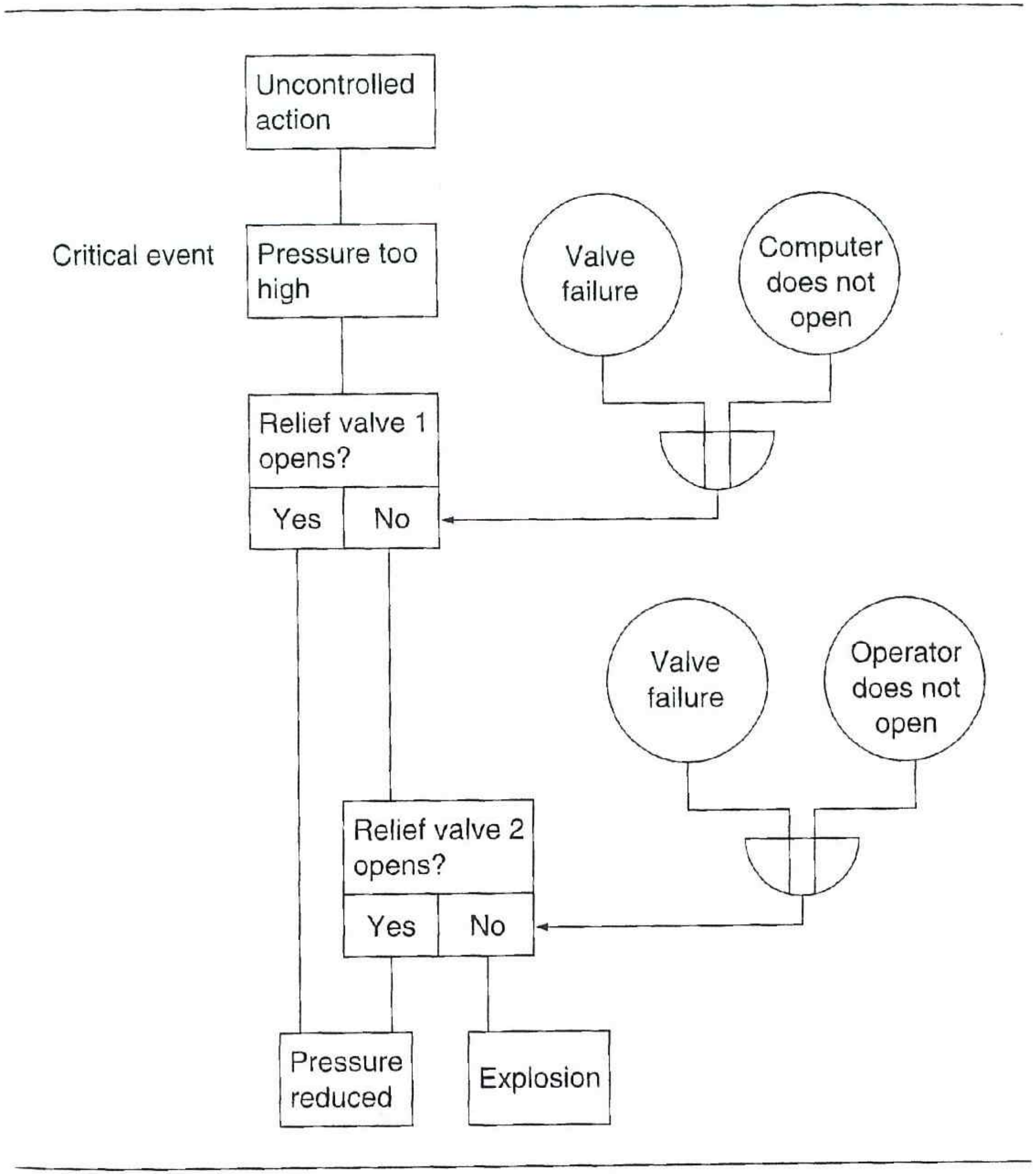


FIGURE 14.6
A cause–consequence diagram.

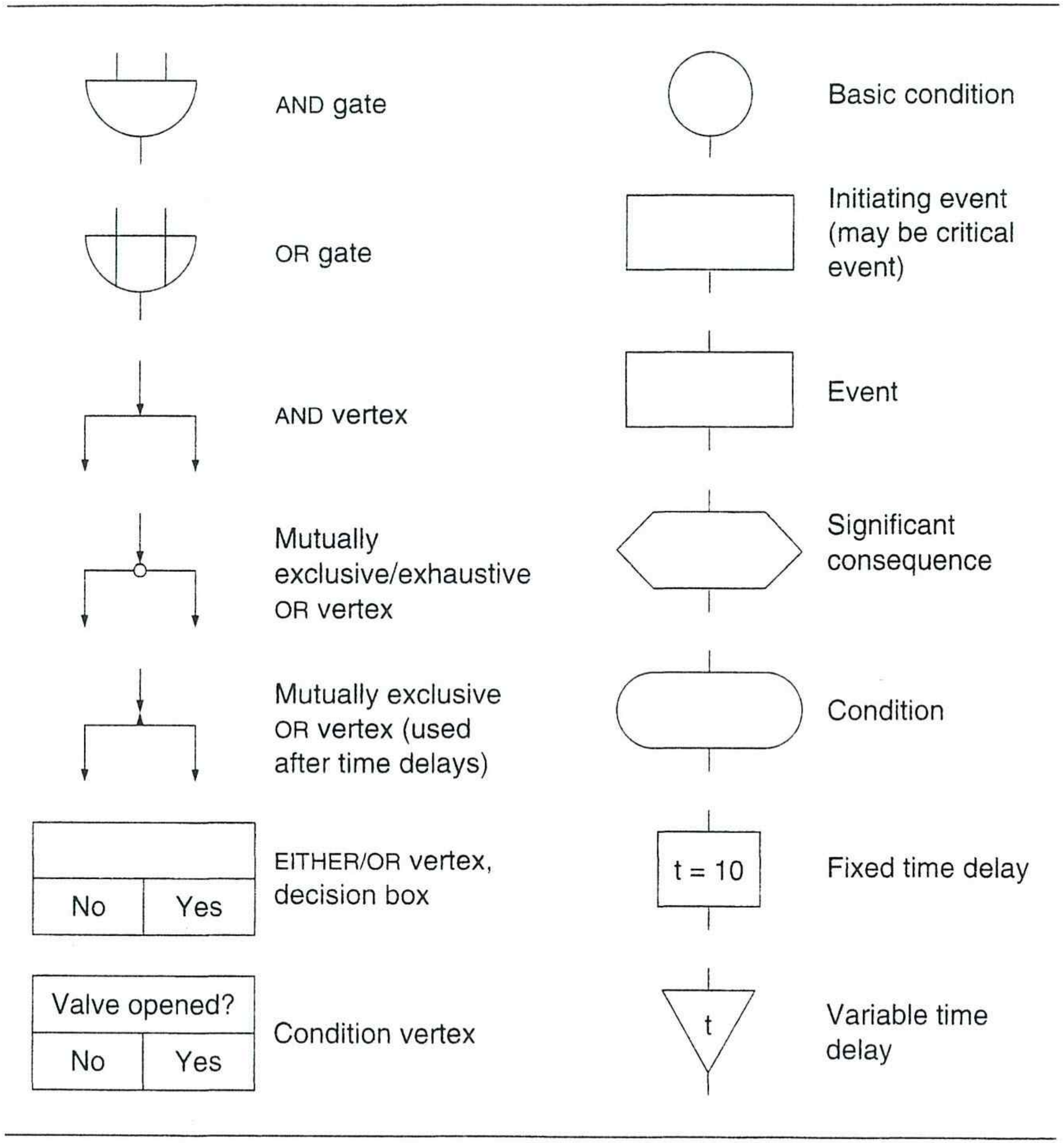

AND gate
OR gate
AND vertex
Mutually exclusive/exhaustive OR vertex
Mutually exclusive OR vertex (used after time delays)
No
Yes
EITHER/OR vertex, decision box
Valve opened?
No
Yes
Condition vertex
Basic condition
Initiating event (may be critical event)
Event
Significant consequence
Condition
t = 10
Fixed time delay
t
Variable time delay

FIGURE 14.7
Cause–consequence diagram symbols.

- CCA
  - Compared to fault trees,
    - Shows the sequence of event explicitly, which makes the diagram especially useful in studying startup, shutdown, and other sequential control problems.
  - Advantage over event trees of allowing
    - The representation of time delays, alternative consequence paths, and combinations of events
    - External conditions and the temporal ordering of events.
  - May be used for quantitative assessment.
- On the negative side,
  - Can become unwieldy.
  - Separate diagrams are required for each initiating event,
  - Outcomes are related only to the cause being analyzed, although they could be caused by other initiating events.
- CCA seems to be used more in Europe than in the United States.

14.7 Hazard and Operability Analysis

- HAZOP is a qualitative technique.
  - Developed in the early 1960s, in England.
  - Chemical industry

  - Purpose is to identify all possible deviations from the design's expected operation and all hazards associated with these deviations.
  - Focus not only on safety but also on efficient operations.

  - Able to elicit hazards in new designs, have not been considered previously.
  - Based on a system theory model of accidents
    - Assumes accidents are caused by deviations from the design or operating intentions.

  - Encourage creative thinking about all the possible ways in which hazards or operating problems might arise.
    - Performed systematically
    - HAZOP team

- A HAZOP team considers
  - The design intention of the plant
  - The potential deviations from the design intention
  - The causes of these deviations from the design intention
  - The consequences of such deviations

- Guidewords are used. (Table 14.1)
  - Questions are generated from the guidewords.
  - A detailed flow chart of the HAZOP process (Figure 14.8)

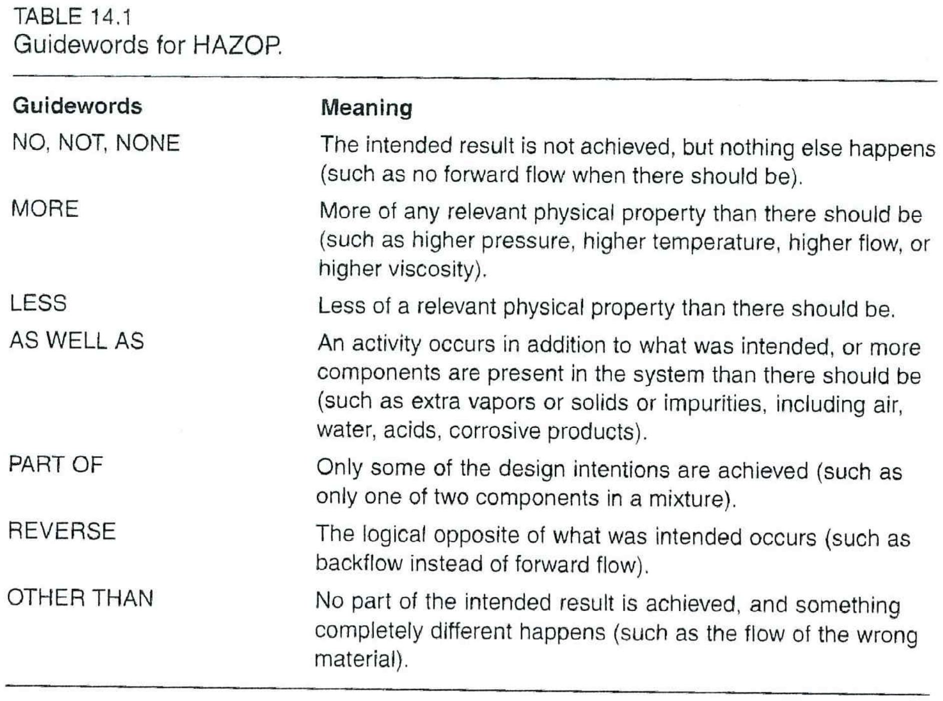
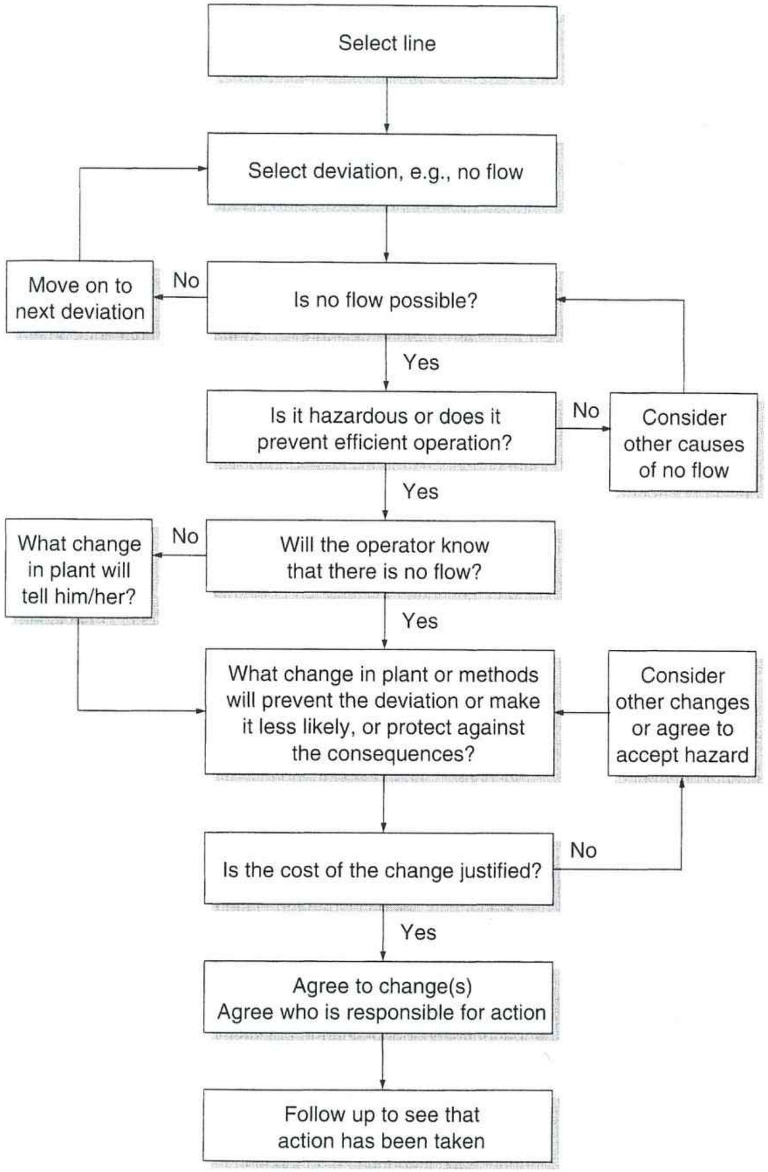


TABLE 14.1
Guidewords for HAZOP.

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).



Select line
Select deviation, e.g., no flow
Move on to next deviation
No
Is no flow possible?
Yes
Is it hazardous or does it prevent efficient operation?
No
Consider other causes of no flow
Yes
What change in plant will tell him/her?
No
Will the operator know that there is no flow?
Yes
What change in plant or methods will prevent the deviation or make it less likely, or protect against the consequences?
Consider other changes or agree to accept hazard
Is the cost of the change justified?
No
Yes
Agree to change(s) Agree who is responsible for action
Follow up to see that action has been taken

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.)

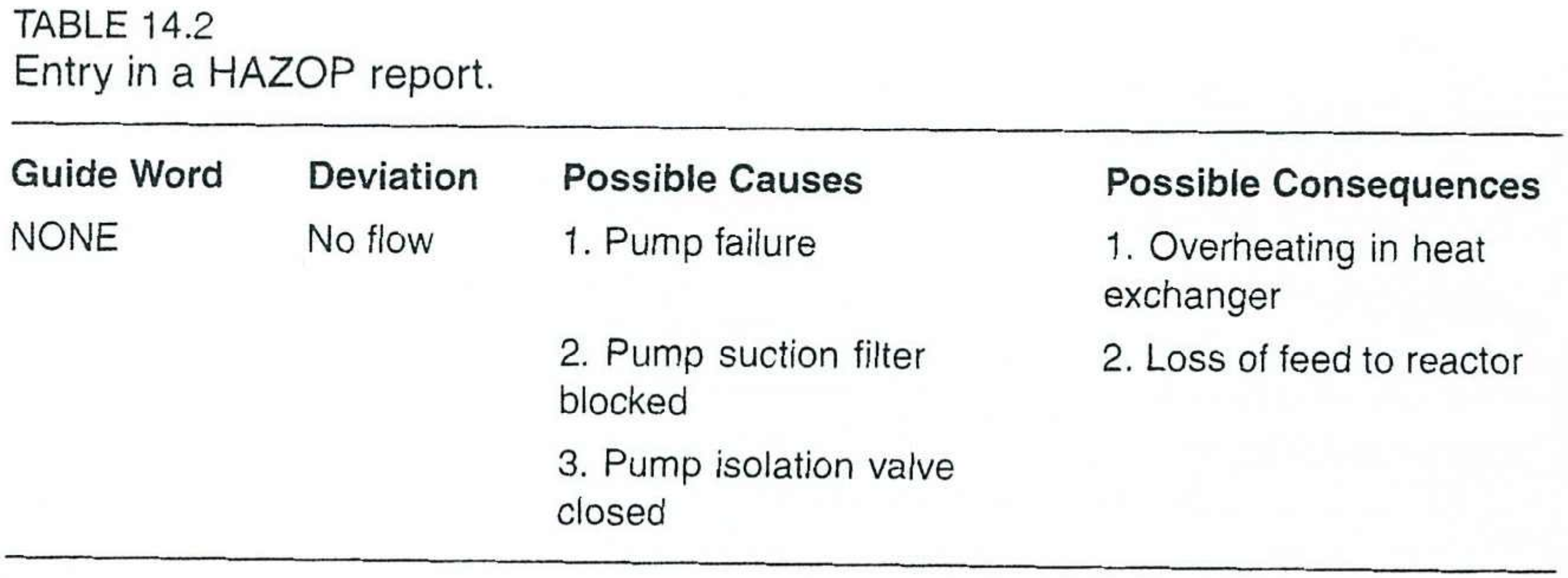


TABLE 14.2
Entry in a HAZOP report.

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	

- HAZOP uses too much information.
  - 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].

- Many companies conduct
  1. Preliminary HAZOPs on conceptual flowcharts and preliminary layout diagrams (Noting only safety aspect, not operating problems)
  2. At a later stage, when the design is almost complete, it may only be possible to reduce the risk by adding devices.
  3. A full HAZOP usually is conducted later in the design process.

- HAZOP
  - Not attempt to provide quantitative results,
  - But instead systematizes a qualitative approach.

  - Strength
    - Simplicity
    - Ease of application
    - Early identification of design problems
    - Potential to find more complex types of hazardous events and causes
    - Encouragement of cross-fertilization of ideas among members of the study team

  - Drawbacks
    - Time and effort required – labor-intensive
    - Limitation imposed by the search pattern
      - Lies very heavily on the judgment of the engineers performing the assessment.
    - Etc.

14.8 Interface Analyses

- Various analysis methods used to evaluate connections and relationships between components,
  - Including incompatibilities and the possibilities for common-cause or common-mode failure.
  - Relationship examined [108]
    - Physical
    - Functional
    - Flow
  - Generally use structural walkthroughs
    - to examine the interface between components and
    - to determine whether a connection provides a path for failure propagation.
      - No output from the unit or interconnection failures
      - Degraded output or partial failures of the unit or interconnection
      - Erratic output (intermittent or unstable operation)
      - Excessive output
      - Unprogrammed output
      - Undesired side effects

- Interface analyses should include connections between components that go through the software.

- Interface analyses are similar to HAZOP, but generalized somewhat.
  - They have the same benefit and limitations.

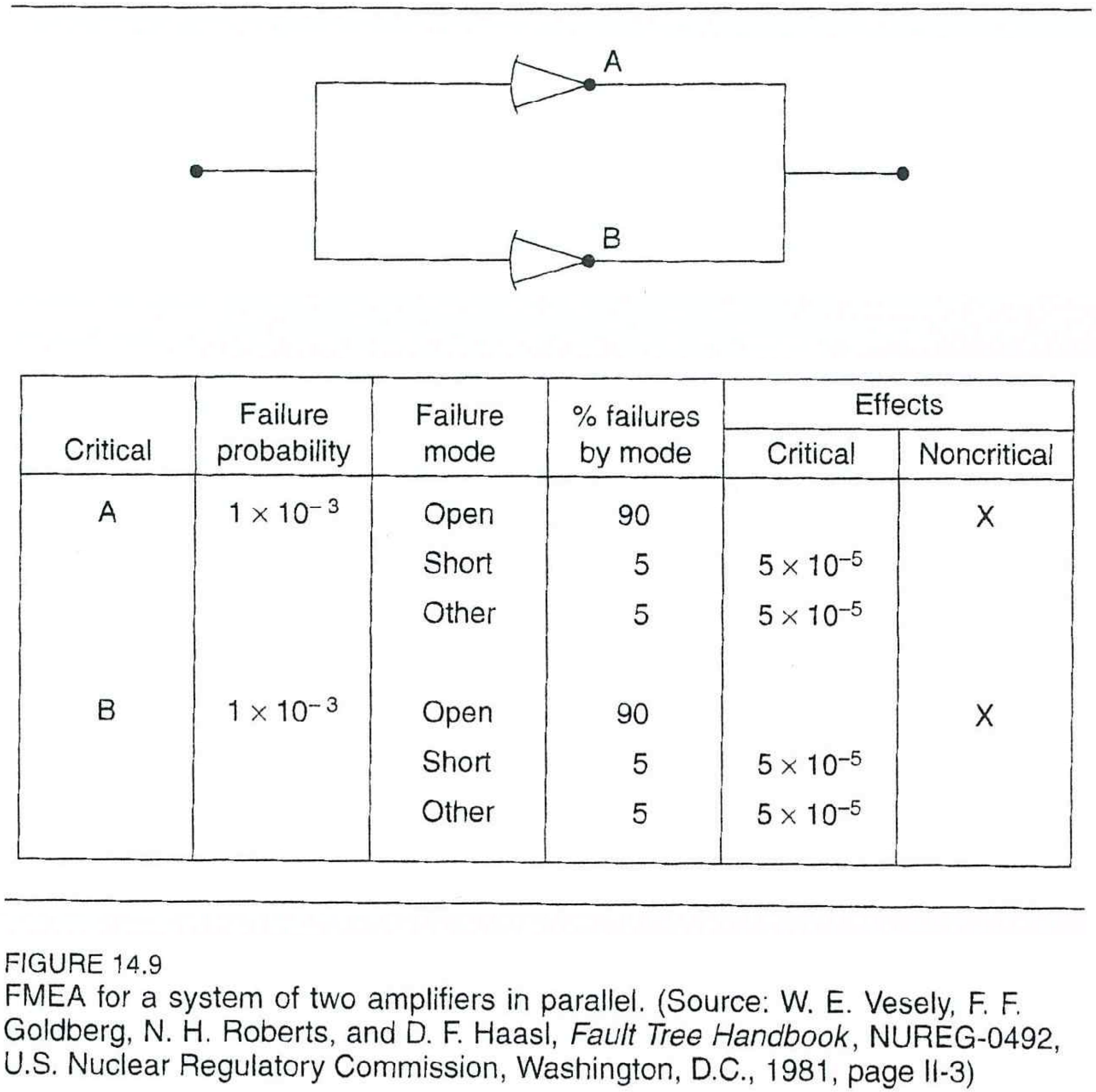
14.9 Failure Modes and Effects Analysis

- Failure Modes and Effects Analysis (FMEA)
  - Developed by reliability engineers to predict equipment reliability.
  - A form (table) of reliability analysis that emphasizes successful functioning rather than hazards and risk.

  - To establish the overall probability that the product will operate without a failure for a specific length of time,
    - Alternatively, the product will operate a certain length of time between failures.

  - Uses forward search based on an underlying chain-of-events models.
    - Where the initiating events are failures of individual components.
      1. Identify and list all components and their failure modes, considering all possible operating modes.
      2. For each failure mode, the effects on all other system components are determined along with the effect on the overall system.
      3. Then, the probability 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.



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)

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

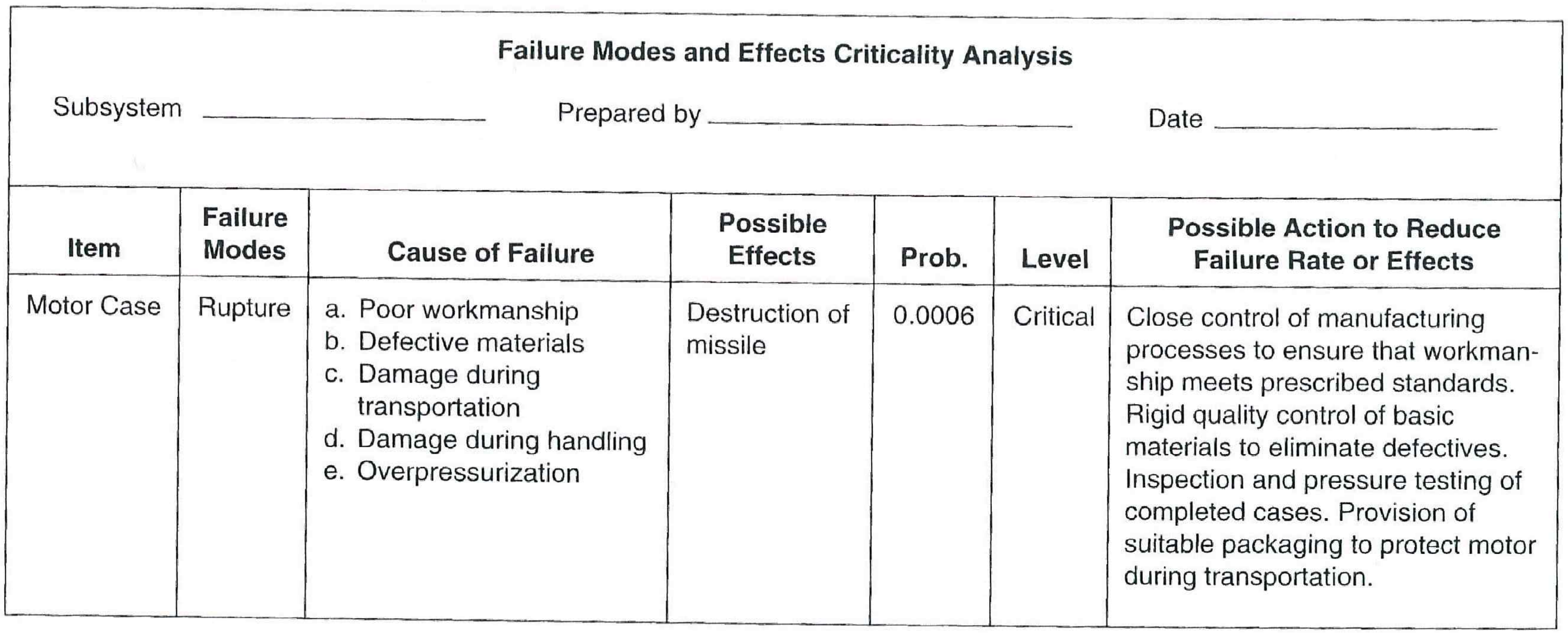
- FMEA
  - Effective for analyzing single units or failures.
  - All the significant failure mode must be known in advance.
    - Systems exhibiting any degree of complexity becomes simply impossible [344].
  - Not provide any systematic approach for identifying failure modes.
  - Not normally consider effects of multiple failures.
    - Each failure is treated as an independent occurrence with no relation to other failures.
    - It becomes simple to apply and the examination is very orderly.
    - But, the results may be of limited use.
  - Pay little attention to human errors in operating procedures, hazardous characteristics of the equipment, or adverse environments [106].

14.10 Failure Modes, Effects, and Criticality Analysis

- Failure Modes, Effects, and Criticality Analysis (FMECA)
  - Basically just an FEMA with a more detailed analysis of the criticality of the failure.
  - Two additional steps added:
    1. The means of control already present or proposed are determined.
    2. The findings are modified with respect to these control procedures.
  - Figure 14.10

- Sometimes, a Critical Item List (CIL) is generated from the results of the FMEA or FMECA.

- The same evaluation applies to FMECA.
  - With the exception that the FMEA does include a description of the means of controlling the failure.



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.

14.11 Fault Hazard Analysis

- Fault Hazard Analysis (FHA)
  - 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 etc.
  - The scope is more restricted since
    - Supposedly only failures that could result in accidents are considered.
  - Two new pieces of information are added about upstream and downstream effects:
    1. Upstream components that could command or initiate the fault in questions
    2. Factors that could lead to secondary failures.
    - The effects on the system are briefly stand in terms of associated damage or malfunction.

- FHA primarily provides guidance on what information to obtain.
  - But it provides no help in actually getting that information.
  - FHA tends to concentrate primarily on single event or failure.

- Hammer says
  - FHA is useful in considering faults that cross organizational interfaces.
  - In others, have little use.

14.12 State Machine Hazard Analysis

- A state machine is a model of
  - The states of a system and the transitions between them. (Figure 14.11)
  - Often used in computer science.

  - Large number of states that these systems have must be specified.
    - State space, abstraction

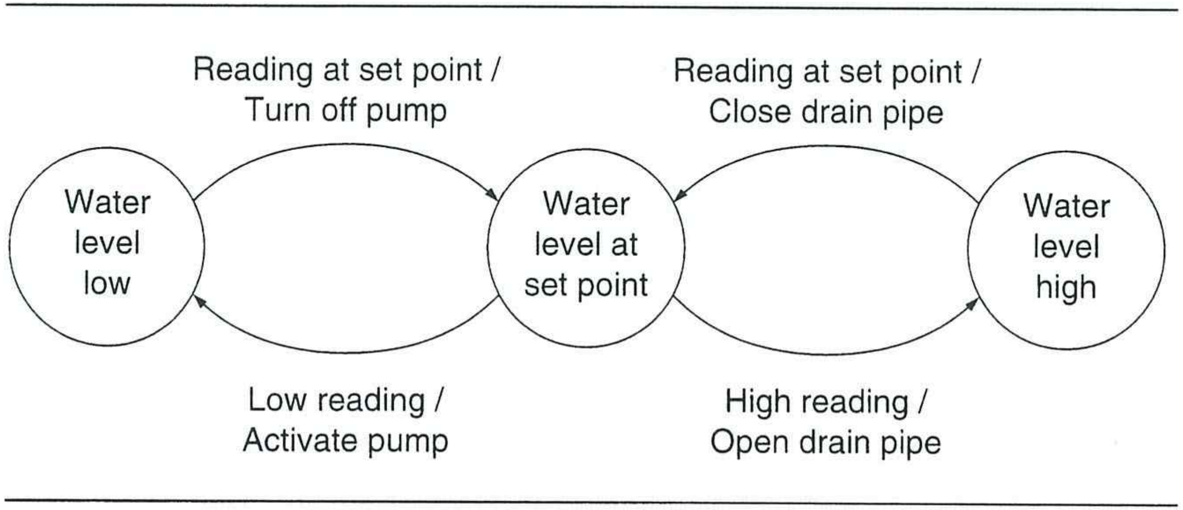

Reading at set point / Turn off pump
Reading at set point / Close drain pipe
Water level low
Water level at set point
Water level high
Low reading / Activate pump
High reading / Open drain pipe

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

- State Machine Hazard Analysis (SMHA) involves
  - Forward search
    - Starts from the initial state of the system, generates all possible paths from that state, and determines whether any of them are hazardous.
    - Unfortunately, the computational effort involved makes this approach impractical, even if computers are used.
  - Backward search
    - Also impractical, in the general sense.

- A practical solution is
  - Start from the hazardous state and only work for enough back along the paths to determine how to change the model to make the hazardous state unreachable [186].

- 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 interface between the software and the hardware.

- Since the model used is formal (that is, it has a mathematical definitions),
  - The analysis procedures can be implemented on a computer.
- SMHA works on a model, not the design itself.
  - It can theoretically be used at any stage of the life cycle.
  - Most effective if performed before the detailed design of the system component begins.
- SMHA's limitations
  - A model must be built, which may be difficult and time consuming.
    - But, it can be used as the system requirements specification.
  - 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.
  - Other types of mathematical models (logic or algebraic models) have been proposed for software.
    - Most have been tried only on very small examples.
    - They are hard to learn and use without an advanced degree in mathematics.
  - 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.

14.13 Task and Human Error Analysis Techniques
14.13.1 Qualitative Techniques

- Some analysis methods for human error have been suggested.
  - Procedure or Task Analysis
  - Operator Task Analysis
  - Action Error Analysis (AEA)
  - Work Safety Analysis (WSA)

- The goal is
  - To obtain the information necessary to design a human-machine interface that
    - Reduces human behavior leading to accidents
    - Improves operators' ability to intervene successfully to prevent accidents.
  - The focus is the operator's task.

14.13.2 Quantitative Techniques

- When the focus is design,
  - Qualitative or semi-quantitative results are usually adequate to achieve the goals.

- Probabilistic assessment of human error
  - Not very advanced.
  - Some problems were discussed in Chapter 13.
  - This chapter describes the current state of the art.
    - Reader 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.
  - Lee classified (see Chapter 10) tasks into
    - Simple and Vigilance Tasks
    - Complex Control Tasks

14.14 Evaluations of Hazard Analysis Techniques

- Only a few critical evaluations of hazard analysis methods have been performed.
  - Most simply evaluate the structure of the methods.

- Taylor, Suokas, and Rouhiainen, however, have actually performed empirical evaluations.

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.

- 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 last of this book.

Chapter 15

SOFTWARE HAZARD AND REQUIREMENTS ANALYSIS

Introduction

> *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 treated to requirements flaws.
  - Incompleteness in the specified and implemented software behavior

- This chapter describes completeness and safety criteria for software requirements specifications.

15.1 Process Considerations

- The software hazard analysis will be influenced by the underlying accident model being used and its assumptions about the contribution of computers to accidents.

- The tasks of the software safety process defined in Section 12.1.1 include:
  1. Trace identified system hazards to the software-hardware interface.
     Translate the identified software-related hazards into requirements and constraints on software behavior.
  2. Show the consistency of the software safety constraints with the software requirements specification.
     Demonstrate the completeness of the software requirements, including the human-computer interface requirements, with respect to system safety properties.

- The first step of the process can be accomplished with top-down or bottom-up hazard analysis.

- Top-down hazard analysis
  - Traces system hazard down to and into the subsystems.
  - Software-related hazards are identified and traced into the software requirements and design.
    - Fault tree analysis down to the software interface

- Bottom-up subsystem hazard analysis
  - Its practicality is limited by the large number of ways that computers can contribute to system hazards.

- Readability and reviewability will be enhanced by
  - Using languages that allow building models that are semantically close to the user’s mental model of the system. (semantic gap)
  - Ideally, the specification language should reflect the way that engineers and application experts think about the system, not the way mathematicians do.

- The second step of the process are
  - To document the identified software behavioral requirements and constraints
  - To show that the software requirements specification satisfies them.
  - Including demonstrating the completeness of the software requirements specification with respect to general system safety properties.
- Most current software hazard and requirements analyses are done in an ad hoc manner.
  - This chapter examines what needs to be accomplished in such an analysis.

15.2 Requirements Specification Components

- Requirements specifications have three components:
  (1) Basic functions or objectives
  (2) Constraints on operating conditions
  (3) Prioritized quality goals to help make tradeoff decision

- Constraints
  - Define the range of conditions within which the system may operate while achieving its objectives.
  - Limit the set of acceptable designs to achieve the objective.
    - Quality considerations
    - Physical limitations of the equipment
    - Equipment performance considerations
    - Process characteristics

- Safety may be and often is involved in both functionality requirements and constraints
  - Goals and constraints often conflict.
  - Tradeoffs among functional goals and constraints must be identified and resolved according to priorities assigned to each.

15.3 Completeness in Requirements Specifications

- Completeness or lack of ambiguity
  - The most important property of requirements specifications with respect to safety
  - Desired software behavior must have been specified in sufficient detail to distinguish it from any undesired program that might be designed.

  - In the sense of a lack of ambiguity from the application perspective
    - The specification is incomplete if the system or software behavior is not specified precisely enough, because the required behavior for some events or conditions is omitted or is ambiguous or is subject to more than one interpretation.

- Criteria
  - The rest of this chapter
  - Criteria for completeness of software requirements specifications
    - Useful in detecting incompleteness that is associated with hazards and accidents.
    - A starting point for a safety checklist for requirements specification to which additions may be made as we discover the necessity.

15.4 Completeness Criteria for Requirements Analysis

- Requirements specification
  - Defines the function to be implemented by the computer.
  - Describes the required black-box behavior of the component.

- Safety analysis, here is concerned only with the black-box behavior of the software.
  - The only aspect of the specification that can directly affect system hazards.

- In this chapter, the control function is described using a state machine model. (Figure 14.11)
  - The criteria are described in terms of the components of a state machine model.
  - The controller outputs to actuators are associated with state changes in the model, which are triggered by measurements of process variables. (Figure 15.1)

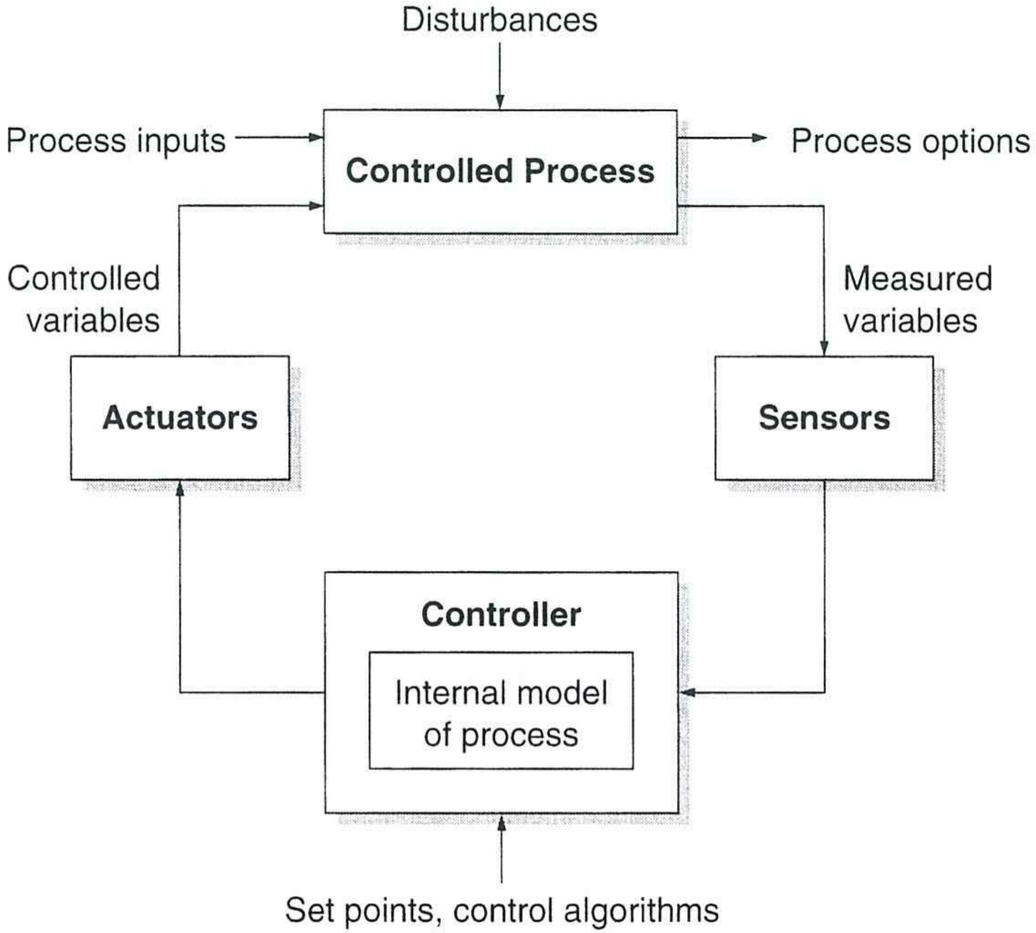


FIGURE 15.1
A black-box requirements specification captures the controller's internal model of the process. Accidents occur when the internal model does not accurately reflect the state of the controlled process.

- Theoretical control laws are defined using the *true* value of the process state.
  - At any time, however, the controller has only *measured* values, which may be subject to time lags or measurement inaccuracies.
- Considering the problems of measurement error and time lags is essential in developing safe control software.

- A state machine model is an abstraction.
  - It models the view of the process maintained by the computer, which is necessarily incomplete.

  - Hazards and accidents can result from mismatches between the software view of the process and the actual state of the process.
    - The model of the process used by the software gets out of synch with the real process.
  - The mismatch occurs because the internal model is incorrect or incomplete or the computer does not have accurate information about the process state.

- Safety depends on the completeness and accuracy of the software (internal) model of the process.

- The goal of completeness analysis is basically
  - To ensure that the model of the process used by the software is sufficiently complete and accurate that hazardous process states do not occur.

  - In response to a single occurrence of the given stimulus or trigger, the program must produce only a single output set.
    - *trigger → output*

  - Not only must the output be produced given a particular trigger, but it must *not* be produced without the trigger:
    - *trigger ← output*

- The next sections
  - Informally describe what is required for a complete specification of the triggers and outputs, and the other parts of a black-box state machine model of software behavior.

15.4.1 Human-Computer Interface Criteria

- The human-computer interface has many possible completeness criteria.
  - Can be formed in terms of high-level abstractions applicable to this interface.
  - Alert queue by Jaffe [135]
  - Transaction
  - Labels

- For human-computer interface queues in general, the requirements specification will include:
  - Specification of the events to be queued
  - Specification of the type and number of queues to be provided
  - Ordering scheme within the queue
  - Operator notification mechanism for items inserted in the queue
  - Operator review and disposal commands for queue entries
  - Queue entry deletion

- Transactions
  - May have multiple events associated with it.
  - Multiple-event transactions require additional completeness criteria such as those to deal with preemption in the middle of a transaction.

  - In general, Jaffe identifies three questions that must be answered in the requirements specification for every data item displayable to a human:
    1. What events cause this item to be displayed?
    2. Can and Should the display of this item ever be updated once it is displayed? If so, what events should cause the update? Events that trigger updates may be
       - External observables
       - The passage of time
       - Actions taken by the viewing operator
       - Actions taken by other operator (in multi-person systems)
    3. What events should cause this data display to disappear?

- Labels
  - The computer may control the labels associated with operator actions.
  - Not only can these labels change, but the software may be responsible for such things as highlighting or deleting under current conditions.

15.4.2 State Completeness

- The operational states can be separated into normal and non-normal processing modes.
  - Completeness criteria can be applied to the transition between these.
  - The system and software must start in a safe state. Interlock should be initiated or checked to be operational at system startup, including startup after temporarily overriding interlocks.
  - The internal software model of the process must be updated to reflect the actual process state at initial startup and after temporal shutdown.
  - All system and local variables must be properly initialized upon startup, including clocks.

- The behavior of the software with respect to inputs received before startup, after shutdown, or when the computer is temporarily disconnected from the process must be specified, or it must be determined that this information can be safely ignored, and this conclusion must be documented.

- Paths from fail-safe (partial or total shutdown) states must be specified. The time in a safe but reduced-function state should be minimized.

- Interlock failures should result in the halting of hazardous functions.

- There must be a response specified for the arrival of an input in any state, including intermediate states.

15.4.3 Input and Output Variable Completeness

- At the black-box boundary, only time and value are observable by the software.
  - Therefore, the triggers and outputs must be defined only as constants or as the value and time of observable events or conditions.
  - All information form the sensors should be used somewhere in the specification.
  - Legal output values that are never produced should be checked for potential specification incompleteness.

15.4.4 Trigger Event Completeness

- The behavior of the control subsystem (computer) is defined with respect to assumptions about the behavior of the other parts of the system.
  - A *robust* system will detect and respond appropriately to violations of these assumptions (such as unexpected inputs)
  - By definition, the robustness of the software built from the specification depends upon the completeness of the environmental assumptions.
  - Thus, completeness of the environmental assumptions is related to the completeness of the specification of the trigger events and the response of the computer to any potential inputs.

- Documenting all environmental assumptions and checking them at runtime may seem expensive and unnecessary.
  - Critical assumptions should be checked at runtime.
  - Even when real-time response is not required, software or hardware should log violations of assumptions for off-line analysis.

15.4.4.1 Robustness Criteria

- To be robust, the events that trigger state changes must satisfy following:
  1. Every state must have a behavior (transition) defined for every possible input.
  2. The logical OR of the conditions on every transition out of any state must form a tautology.
  3. Every state must have a software behavior (transition) defined in case there in no input for a given period of time (a timeout).

  - Guarantee that if there is a trigger condition for a state to handle inputs within a range, there will some transition defined to handle data that is out of range.
  - Also requirements for a timeout

- The use of “Otherwise” clause is not appropriate for safety-critical systems.
  - It is better to explicitly delineate exactly what cases provide the “otherwise” condition and then check for tautological completeness [135].

15.4.4.2 Nondeterminism

- Another restriction can be placed on the transition events to require deterministic behavior:
  - The behavior of the state machine should be deterministic (only one possible transition out of a state is applicable at any time.)

15.4.4.3 Value and Timing Assumptions

- Completeness depends upon the *amount* and *type* of information (restrictions and assumptions such as legal range) that is included in the triggers.
  - The more assumptions about the triggers included, the more likely that the four above criteria will ensure that the requirements include response to unplanned events.
- Many assumptions and conditions are application dependent.
- However, some types of assumptions are essential and should always be specified for all inputs to safety-critical systems.
  - In real-time system, the times of inputs and outputs are as important as the values.
- Therefore, both value and time are required in the characterization of the environmental assumptions (triggers) and in the outputs.
  - Essential Value Assumptions
  - Essential Timing Assumptions

- Essential Value Assumptions
  - States the values or ranges of values of the trigger variables and events
  - All incoming values should be checked and a response specified in the event of an out-of-range or unexpected value.
- Essential Timing Assumptions
  - Timing problems are often a common cause of runtime failures.
  - Time is often inadequately specified for software.
  - Two different timing assumptions are essential in the requirements specification of triggers
    - Timing intervals
    - Capacity or load
- Timing intervals
  - All inputs must be fully bounded in time, and the proper behavior specified in case the limits are violated or an expected input does not arrive.
  - A trigger involving the nonexistence of an input must be fully bounded in time.

- Capacity or Load
  - A minimum and maximum load assumption must be specified for every interrupt-signaled event whose arrival rate is not dominated (limited) by another type of event.
  - A minimum-arrival-rate check by the software should be required for each physically distinct communication path. Software should have the capacity to query its environment with respect to inactivity over a given communication path.
  - The response to excessive inputs (violations of load assumptions) must be specified.
  - If the desired response to an overload condition is performance degradation, the specified degradation should be graceful and operators should be informed.
  - If function shedding or reconfiguration is used, a hysteresis delay and other checks must be included in the conditions required to return to normal processing load.

15.4.5 Output Specification Completeness

- As with trigger events, the complete specification of the behavior of an output event requires both its value and its time.
  - Safety-critical outputs should be checked for reasonableness and for hazardous values and timing.
  - Checking to make sure that output values are legal or reasonable is straightforward and helpful in detecting software or other errors.
    - Environmental Capacity Considerations
    - Data Age
    - Latency

- Environmental Capacity Considerations
  - For the largest interval in which both input and output loads are assumed and specified, the absorption rate of the output environment must equal or exceed the input arrival rate.
  - Contingency action must be specified when the output absorption rate limit will be exceeded.
  - Update timing requirements or other solution to potential overload problems, such as operator event queues, need to be specified.
  - Automatic update and deletion requirements for information in the human-computer interface must be specified.
  - The required disposition for obsolete queue events must include specification of what to do when the event is currently being displayed and when it is not.

- Data Age
  - All inputs used in specifying output events must be properly limited in the time they can be used (data age). Output commands that may not be able to be executed immediately must be limited in the time they are valid.
  - Incomplete hazardous action sequences (transactions) should have a finite time specified after which the software should be required to cancel the sequence automatically and inform the operator.
  - Revocation of a partially completed action sequence may require (1) specification of multiple times and conditions under which carrying automatic cancellation or postponement actions are taken without operator confirmation and (2) specification of operator warnings to be issued in the event of such revocation.

- Latency
  - A latency factor must be included when an output is triggered by an interval of time without a specified input and the upper bound on the interval is not a simple, observable event.
  - Contingency action may need to be specified to handle events that occur within the latency period.
  - A latency factor must be specified for changeable human-computer interface data displays used for critical decision making. Appropriate contingency action must be specified for data affecting the display that arrives within the latency period.
  - A hysteresis delay action must be specified for human-computer interface data to allow time for meaningful human interpretation. Requirements may also be needed that state what to do if data should have been changed during the hysteresis period.

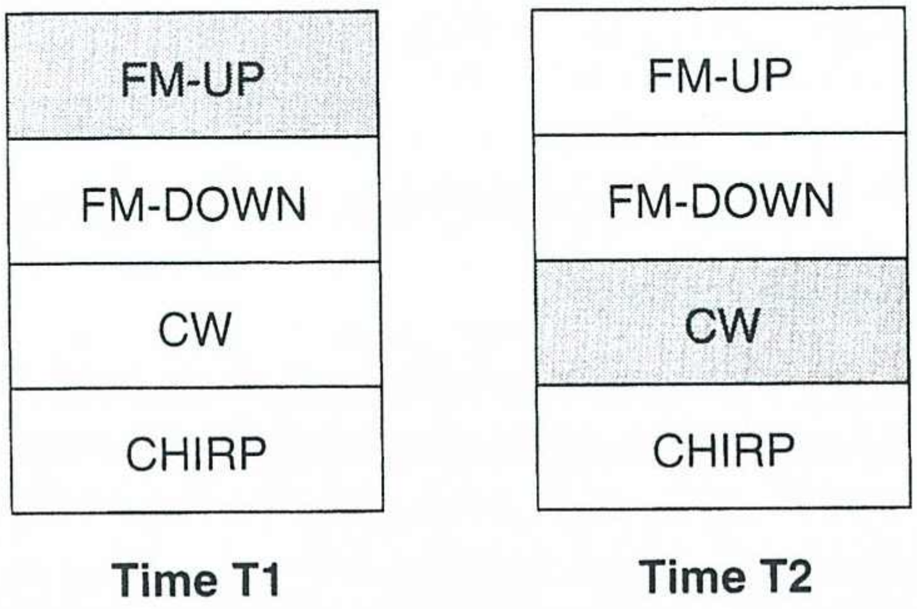


FIGURE 15.2
Two consecutive snapshots of an operation action menu with the recommended action highlighted. The recommendation must be constant long enough for meaningful human interpretation. Requirements are also needed to deal with latency problems when the recommended action changes.

15.4.6 Output to Trigger Event Relationships

- Some criteria for analyzing requirements specifications relate not to input or output specifications alone, but to the relationship between them.
  - Basic feedback loops, as defined by the process control function, must be included in the software requirements. That is, there should be an input that the software can use to detect the effect of any output on the process. The requirements must include appropriate checks on these inputs in order to detect internal or external failures or errors.
  - Every output to which a detectable input is expected must have associated with it: (1) a requirement to handle the normal response and (2) requirements to handle a response that is missing, too late, too early, or has an unexpected value.
  - Spontaneous receipt of a nonspontaneous input must be detected and responded to as an abnormal condition.
  - Stability requirements must be specified when the process is potentially unstable.

15.4.7 Specification of Transition Between States

- Requirements analysis may involve examining the paths between states.
  - These paths are uniquely defined by the sequence of trigger events along the path.
  - Transitions between modes are particularly hazardous and susceptible to incomplete specification, and they should be carefully checked.

  - Reachability
  - Recurrent Behavior
  - Reversibility
  - Preemption
  - Path Robustness

- Reachability
  - All specified states must be reachable from the initial state.

- Recurrent Behavior
  - Desired recurrent behavior must be part of at least one cycle. Required sequences of events must be implemented in and limited by the specified transitions.
  - States should not inhibit the production of later required outputs.

- Reversibility
  - Output commands should usually be reversible.
  - If x is to be reversible by y, there must be a path between the state where x is issued and a state where y is issued.

- Preemption
  - Preemption requirements must be specified for any multistep transactions in conjunction with all other possible control activations.

- Path Robustness
  - Soft and hard failure modes should be eliminated for all hazard-reducing outputs. Hazard-increasing outputs should have both soft and hard failure modes.
  - Multiple paths should be provided for state changes that maintain or enhance safety. Multiple inputs or triggers should be required for paths from safe to hazardous satets.

15.5 Constraint Analysis

- Requirements must also be shown to include the indentified, system-specific safety requirements and to be consistent with the identified software system safety constraint.
  - Transitions must satisfy software system safety requirements and constraints.

- In general, software-related hazards involve
  - Failing to perform a required function: The function is never executed or no answer is produced.
  - Performing an unintended (unrequired) function, getting the wrong answer, issuing the wrong control instruction, or doing the right thing but under inappropriate conditions.
  - Performing functions at the wrong time or in the wrong order.
  - Failing to recognize a hazardous condition requiring corrective action.
  - Producing the wrong response to a hazardous condition.

- Constraint analysis on the software requirements specification includes a reachability analysis to determine whether the software, as specified, could reach the identified hazardous states.
  - Reachability hazardous states should be eliminated or, if that is not possible (they are needed to achieve the goals of the system), their frequency and duration reduced.

- More generally, the specification may be checked for a general safety policy that is defined for the particular system.

- It may not be possible to build a complete safe system.
  - In those hazardous events, the system must be redesigned or abandoned, or some risk muse be accepted.
  - This risk can be reduced by providing procedures to
    - Minimize the probability of the hazardous state leading to an accident
    - Minimize the effects of an accident.

15.6 Checking the Specification Against the Criteria

- The actual procedures that can be used to analyze a particular requirements specification will depend on the form of that specification.
  - Criteria for completeness of states, inputs and outputs, and the relationship between inputs and outputs are easily checked for any type of specification.
  - Criteria for the transitions between states will be checkable to a greater or lesser extent depending on the formality of the specification, the size of the specification, and the availability of software tools to help with the checking.

- RSML [116,117]
  - Automated the checking of the robustness and nondeterminism criteria for specification written in RSML.
  - Additional tools were created for safety analysis of RSML requirements specifications.

- Some criteria can be enforced by using a specification language that incorporates enforcement in its syntax.

- On many projects, requirements are not complete before software development begins.
  - To avoid costly redesign and recoding, the requirements specification and analysis should be as complete as possible, as early as possible.

Chapter 16

DESIGN FOR SAFETY

Introduction

Engineers should recognize that reducing risk is not an impossible task, even under financial and time constraints. All it takes in many cases is a different perspective on the design problem.

- Mike Martin and Roland Schinzinger
Ethics in Engineering

Software temptations are virtually irresistible. The apparent ease of creating arbitrary behavior makes us arrogant. We become sorcerer's apprentices, foolishly believing that we can control any amount of complexity. Our systems will dance for us in ever more complicated ways. We don't know when to stop… We would be better off if we learned how and when to say no.

-G.F. McCormick
-When Reach Exceeds Grasp

- Safety must be designed into a system.
  - The information obtained in the hazard analysis needs to be used in the design.
- Often, the most complex and tricky problems are left to the operators.
  - Now the operators are being replaced by computer software that is supposed to carry out the same procedure, the onus will be on the software.
- Poorly designed risk reduction measures can actually increase risk and cause accidents.
  - In some cases, safety mechanisms are an attempt to compensate for a poor basic system design.
- Software (Computer) now controls dangerous systems.
  - Software engineers need to understand the basic principles behind safe system design so they can include them in the software design.
  - The use of computers also introduces new possibilities for system safety in terms of increased functionality and more powerful protections mechanisms.
  - The system must be protected against software errors.

- Unfortunately, protecting against software errors may be more difficult than protecting against hardware failures.
  - It is difficult to plan for software errors, since they are unpredictable.
  - Many computer hardware failures and some software errors can be detected and handled, but doing so requires a great deal of planning and efforts.

- This chapter
  - Describes standard system safety design approaches and some of their applications to software design.
  - Hopefully, they will start people thinking about additional ways to apply standard safety engineering approaches to software design.

16.1 The Design Process

- Two basic approaches to safe design
  - (1) Applying standards and codes of practice that reflect lessons learned from previous accidents.
  - (2) Guiding design by hazard analysis

  - These approaches are complementary and both should be used.

16.1.1 Standards, Codes of Practice, and Checklists

- For hardware,
  - General safety design principles have been incorporated into standards, codes of practice, and checklists in order to pass on lessons learned from accidents.

- For software,
  - There are no equivalent standards for safe software.

- Design checklists
  - Another way to systemize and pass on engineering experience and knowledge.
  - Table 16.1 gives examples of partial checklists for mechanical hazards and pressure systems.
  - Usually focus on those that lead to known hazards.
  - Software is not by itself hazardous, there are no generic software hazards to consider in design checklists.
    - But, checklists could be constructed that included safe software design features. (as opposed to common coding errors)

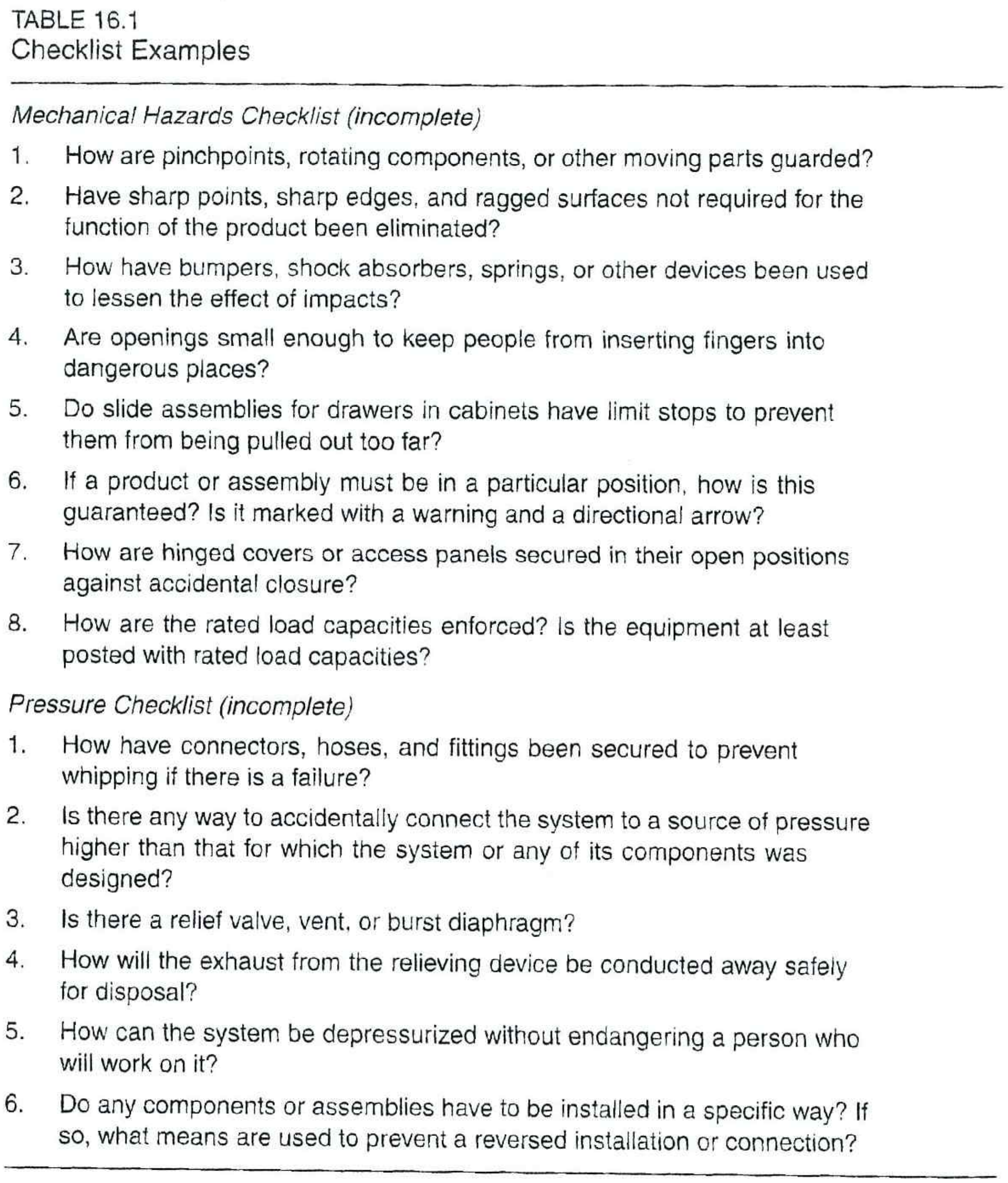


TABLE 16.1
Checklist Examples

Mechanical Hazards Checklist (incomplete)

1. How are pinchpoints, rotating components, or other moving parts guarded?
2. Have sharp points, sharp edges, and ragged surfaces not required for the function of the product been eliminated?
3. How have bumpers, shock absorbers, springs, or other devices been used to lessen the effect of impacts?
4. Are openings small enough to keep people from inserting fingers into dangerous places?
5. Do slide assemblies for drawers in cabinets have limit stops to prevent them from being pulled out too far?
6. If a product or assembly must be in a particular position, how is this guaranteed? Is it marked with a warning and a directional arrow?
7. How are hinged covers or access panels secured in their open positions against accidental closure?
8. How are the rated load capacities enforced? Is the equipment at least posted with rated load capacities?

Pressure Checklist (incomplete)

1. How have connectors, hoses, and fittings been secured to prevent whipping if there is a failure?
2. Is there any way to accidentally connect the system to a source of pressure higher than that for which the system or any of its components was designed?
3. Is there a relief valve, vent, or burst diaphragm?
4. How will the exhaust from the relieving device be conducted away safely for disposal?
5. How can the system be depressurized without endangering a person who will work on it?
6. Do any components or assemblies have to be installed in a specific way? If so, what means are used to prevent a reversed installation or connection?

- With the introduction of computers into safety-critical systems,
  - Many of the lessons learned and incorporated into hardware standards and codes are being lost.

  - Some design principles may not hold when computers replace electro-mechanical systems.
    - But most do, and there must be incorporated into system and software design to avoid needless repetition of past accidents.

16.1.2 Design Guided by Hazard Analysis

- Although checklists and standards are extremely important, their use alone is not adequate to prevent accidents,
  - Especially, complex systems or in systems where computers allow new features that are not necessarily handled by proven designs and standards.
  - The design must also eliminate or control the specific hazards identified for a particular system.

- Software design tasks (in Chapter 12)
  - Develop system-specific software design criteria and requirements, testing requirements, and computer-human interface requirements based on the identified software safety constraints and software-related hazards.
  - Trace safety requirements and constraints to the code; identify those parts that control safety-critical functions and any safety-critical paths that lead to their execution. Design to control the identified hazards.

- System hazard analysis identifies software-related safety requirements and constraints, which are used to validate the software requirements.

- The first step in using hazard analysis information in design
  - To generate design criteria and general principles for the software.
  - From the start, testability and analyzability of the design should receive serious consideration in decision making.
  - A traceability matrix and tracking procedures within the configuration control system need to be established to ensure traceability of safety requirements and their flow through the documentation.
  - Conditions change, and decision need to be reviewed periodically.

16.2 Types of Design Techniques and Precedence

- Increased emphasis on preventing accidents instead of the more standard engineering reliance on learning from our failures.

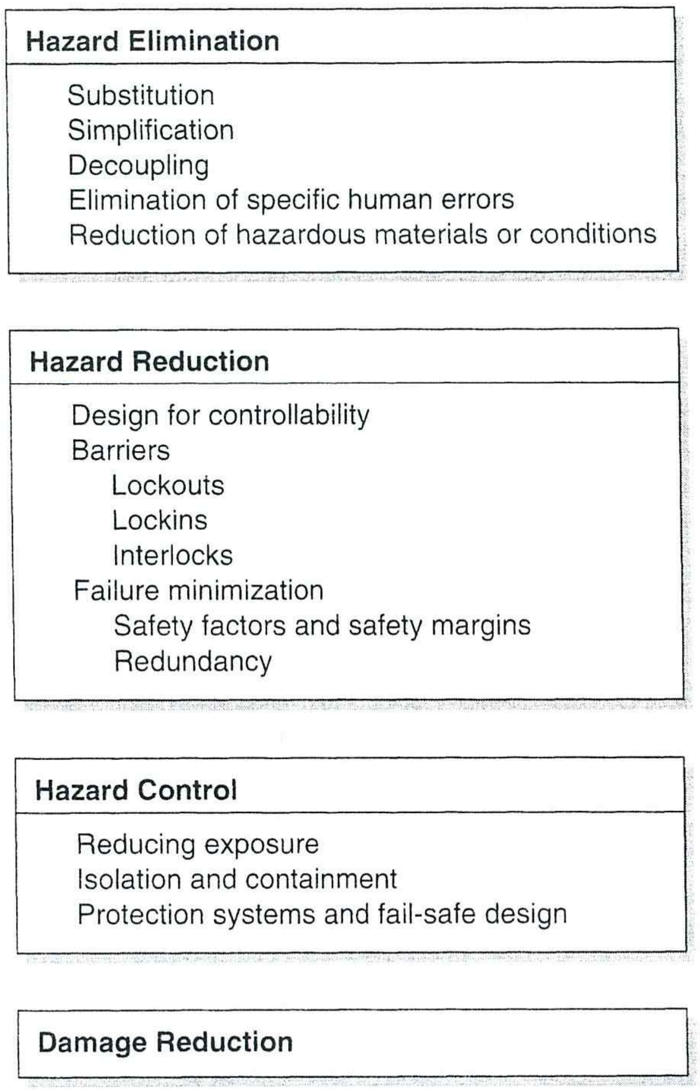
- The basic system safety design goal
  - To eliminate identified hazards or
  - If that is not possible, to reduce the associated risk to an acceptable level.

- Risk is a function of
  (1) The likelihood of a hazard occurring
  (2) The likelihood of the hazard leading to an accident (including duration and exposure)
  (3) The severity or consequences of the accident (see Chapter 9)

  - A design can be made safer by reducing any or all of these three factors.

- System safety guidelines suggest that risk reduction procedures be applied with the following precedence:
  1. Hazard elimination
  2. Hazard reduction
  3. Hazard control
  4. Damage minimization

- Figure 16.1
  - The basic system safety design techniques and their precedence within the general categories.
  - As seen in Chapter 10, many different models of accidents have been proposed.
  - The specific design features applied will therefore depend upon the accident model selected.



Hazard Elimination

- Substitution
- Simplification
- Decoupling
- Elimination of specific human errors
- Reduction of hazardous materials or conditions

Hazard Reduction

- Design for controllability
- Barriers
  - Lockouts
  - Lockins
  - Interlocks
- Failure minimization
  - Safety factors and safety margins
  - Redundancy

Hazard Control

- Reducing exposure
- Isolation and containment
- Protection systems and fail-safe design

Damage Reduction

FIGURE 16.1
Safe design techniques in order of their precedence.

16.3 Hazard Elimination

- The most effective way to deal with a hazard
- Several techniques can be used to achieve an intrinsically safe design:
  - Substitution
  - Simplification
  - Decoupling
  - Elimination of the potential for human errors
  - Reduction of hazardous materials or conditions

16.3.1 Substitution

- Substitute safe or safer conditions or materials.
  - Substituting nonflammable materials for combustible materials
  - Substituting nontoxins for toxins.
- Substitution may introduce other hazards,
  - But the goal is for these new hazards to be minor.
- Several examples in the text

16.3.2 Simplification

- One of the most important aspects of safe design is simplicity.

- A simple design tends
  - To minimize the number of parts, functional nodes, and interfaces
  - It has a small number of unknowns in terms of interactions with the system and with the operations [265].

- Perrow [259] examined a large number of accidents and concluded that
  - Interactive complexity and tight coupling are the major common factors.

- William Pickering, a director of Jet Propulsion Lab. Said [256]
  - "The most conservative designs capable of fulfilling the mission requirements must be considered. Conservative design involves, wherever possible, the use of flight-proven hardware and, for new designs, the application of state-of-the-art technology, thereby minimizing the numbers of unknowns present in the design. New designs and new technologies are utilized, but only when already existing flight-proven designs cannot satisfy the mission requirements, and only when the new designs have been extensively tested on the ground."

- Simpler systems provide fewer opportunities for error and failure.
  - When the interactions reach a level of complexity where they cannot be thoroughly planned, understood, anticipated, and guarded against, then accidents occur [259].

- Interfaces are a particular problem in safety.
  - Design errors are often found in the transfer of functions across interfaces.
  - Simple interfaces help to minimize such errors and to make the designs more testable.
  - Reducing and simplifying interfaces will reduce risk.
    - Interface problems often lie in the control systems.
    - A basic design principle is that control systems not be split into pieces [344].

- Some of the reasons for complexity in system design [152]
  - The need to add on complicated equipment to control hazards
  - A desire for flexibility
  - The use of some types of redundancy to increase realiability

- Adding computers to the control of systems often results in increased complexity.
  - Adding new functions to a system using computers is relatively easy.
  - Even when there are failures, they are often attributed to factors other than the inherent complexity of the project attempted.

- Many software designs are unnecessarily complex.
  - TNondeterminism in many popular software design techniques is inherently unsafe.
    - Because of the impossibility of completely testing or understanding all of the interactions and states the software can get into.
  - Nondeterminism also makes diagnosing problems more difficult.
    - Because it makes software errors look like transient hardware faults.

  - The need for deterministic software execution stems from
    (1) The need for time periodicity in control systems
    (2) The need to analyze and predict algorithm behavior
    (3) The need to test software and to reproduce test conditions and replicate events
    (4) The need of the human operator to rely on consistency

- A complex software design is usually easier to build than a simple one.
  - Constructing a simple software design for a nontrivial problem usually requires discipline, creativity, restraint, and time.

- The software design should also eliminate hazardous effects of common hardware failures on the software.

16.3.3 Decoupling

- A tightly coupled system is highly interdependent.
  - A malfunctioning part in a tightly coupled system cannot be easily isolated.
  - Accidents in tightly coupled systems are a result of unplanned interactions.
    - Can cause domino effects that eventually lead to a hazardous system state.

- Computers tend to increase coupling in systems
  - Since they usually control multiple system components.
  - They become the coupling agent unless steps are taken to avoid it.

- Modularization is used to control complexity.
  - To minimize, order, and make explicit the connections between modules.

- Information hiding is used.
  - To encapsulate every module's design decisions that it hides from all other models.

- When the highest design goal is safety, modularization may involve
  - Grouping together the safety-critical functions
  - Reducing the number of such modules as much as possible.

- After safety-critical functions are separated from noncritical functions,
  - The designer needs to ensure that errors in the noncritical modules cannot impede the operation of the safety-critical functions.

- Safety kernel
  - Might consist of a protected set of safety-critical functions
  - Might contain operating system functions that protect the safety-critical modules.

- Firewall
  - Design analysis procedures can be used to identify safety-critical modules and data, which can then be protected by *firewalls*.

16.3.4 Elimination of Specific Human Errors

- Human error is often implicated in accidents.

- Several ways to eliminate human errors
  - To design so that there are few opportunities for error in the operation and support of the system.
    - For example, the design can make incorrect assembly impossible or difficult.

  - The design of a programming language can affect human errors in several ways [128]
    - Masterability, fault proneness, understandability, maintainability, and checkability

  - To write specifications and programs that are easily understood.

16.3.5 Reduction of Hazardous Materials or Conditions

- It may still be possible to reduce the amount of the material to a level where the system operates properly but the hazard is eliminated or significantly reduced.
  - Even if completely eliminating a dangerous material or substituting a safer one is not possible.
    - A plant can be made safer by reducing inventories of hazardous materials in process or storage.
  - Another way is to change the conditions under which the hazardous material is handled.
    - To use hazardous materials under the least hazardous conditions possible while achieving the system goals.

- The principle of reduction of hazardous materials or conditions can also be applied to software.
  - For example, software should contain only the code that is absolutely necessary: Operational software should not contain unused executable code.
  - Processor memory not used by the program should be initialized well.

16.4 Hazard Reduction

- Even if hazards cannot be eliminated, in many cases they can be reduced.

- Safeguard
  - Can be used to limit hazards.
  - Passive safety devices (safeguards)
    - Maintain safety by their presences or fail into safe states
    - Does not require any special action or cooperation to be effective.

  - Active safeguards
    - Requires some actions to provide protection:
      - Detecting a condition (monitoring)
      - Measuring some variables
      - Interpreting the measurement (diagnosis)
      - Responding

  - Active protection which requires that a hazard or condition be detected and corrected is preferable.

16.4.1 Design for Controllability

- One way to reduce hazards is to make the system easier to control, for both humans and computers.
  - Incremental Control
  - Intermediate States
  - Decision Aids
  - Monitoring
  - Monitoring Computers

- Incremental Control
  - Allowing critical actions to be performed incrementally rather than a single step [98].
  - The controller can
    (1) Use feedback from the behavior of the controlled process to test the validity of the models upon which the decisions were made, and
    (2) Take corrective action before significant damage is done.

- Intermediate States
  - A design that gives the operator more options than just continuing to run under the given conditions.
  - Various types of fall-back or intermediate states.
    - Multiple levels of functionalities
      - The software must be designed to handle the multiple control modes and the transitions between them.

- Decision Aids
  - Computers can also be provided to operators to assist in controlling the plant.
  - Checking
    - Checking if process measurements have exceeded their limits and an alarm needs to be raised.
    - In nuclear industry
      - Automated alarm analysis [10]
      - The computer may structure the alarms as trees or networks showing the interconnections between the various alarms in the plant to help the operator diagnose the problem. (see Chapter 17)
  - Analyzing process disturbance
    - Collects information, validate the data, filter and derive process variables from measured variables, and separate noise from true deviations.
    - Quite difficult to perform reliably and controversial from a safety standpoint.
  - Valve sequencing programs
    - Analyzing a valve sequence proposed by an operator to determine whether it is hazardous
    - Synthesizing safe valve sequences [172].

- Monitoring
  - Detecting a problem requires some form of monitoring
    (1) Checking conditions that are assumed to indicate a potential problem.
    (2) Validating or refuting assumptions made during design and analysis.
  - For example,
    - A simulation of the controlled process used to validate the software requirements during development might be executed in parallel with the control software during operations and compared with the actual process measurements.
  - Two ways to detect equipment malfunction [172]
    (1) By monitoring equipment performance
      - Can be made from the control room using instrument display
      - Compare redundant information with the measurement
    (2) By monitoring equipment condition
      - Requires the operator to check the equipment physically

- Monitors, in general, should
  1. (1) Detect problems as soon as possible after they arise and at a level low enough to ensure that effective action can be taken
  2. (2) Be independent from the devices they are monitoring
  3. (3) Add as little complexity to the system as possible
  4. (4) Be easy to maintain, check, and calibrate

- Independence is always limited [9].
  - Checks requires access to the information to be checked.
  - Providing access may introduce the possibility of corrupting the information.
  - Monitoring depends on assumptions about the structure of the system, types of faults, and errors may occur.

- Monitoring system
  - Provides feedback to an automatic device or the operators (or both) so that they can make remedial action.
  - Especially important when performing functions known to be particularly hazardous.
    - Startup, shutdown or any non-normal operating mode
    - When computers are involved,
      - » Checks might include periodic tests of memory, the data bus, data transmission, and inter-CPU communication.

- Monitoring Computers
  - Checking using computers can be classified in a hierarchy.
  - The farther down, the better.
    - However, some errors cannot be detected except at a high level of abstraction.

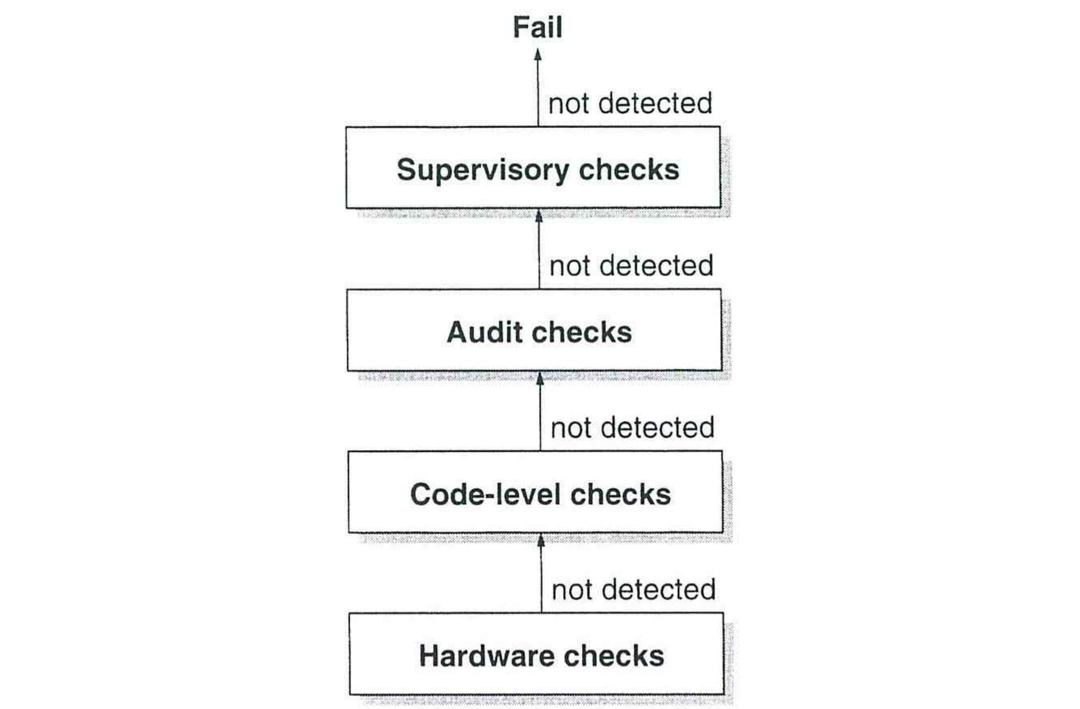


FIGURE 16.2
A hierarchy of software checking.

- Hardware checks
  - Detect computer hardware failures or individual instructions errors
    - Checksums
    - Self-checking is often built into computer hardware. (e.g., parity checks)
    - Facilities for additional checks are or can be included into operating systems.
- Code-level checks
  - Detect coding errors and implementation errors
    - Assertions
- Audit checks
  - Performed by a process separate from that being checked. (independent monitoring)
  - Can check
    - Data passed between modules
    - Consistency of global data structure
    - Expected timing of modules or processes
- Supervisory checks
  - A supervisory system observes the computer externally
  - Provide a viewpoint that is totally detached from the observed system.
  - Additional hardware or completely separated hardware my be used.

- Care needs to be taken to ensure that
  - The added monitoring and checks do not cause failures themselves.
- One way of dealing with these problems is to use a safety kernel or safety executive [184, 185].
  - Coordinates the various monitoring mechanisms.
  - Allows centralization and encapsulation of safety mechanism with the concomitant advantages of reusability and possible formal verification of the operations of the executive.

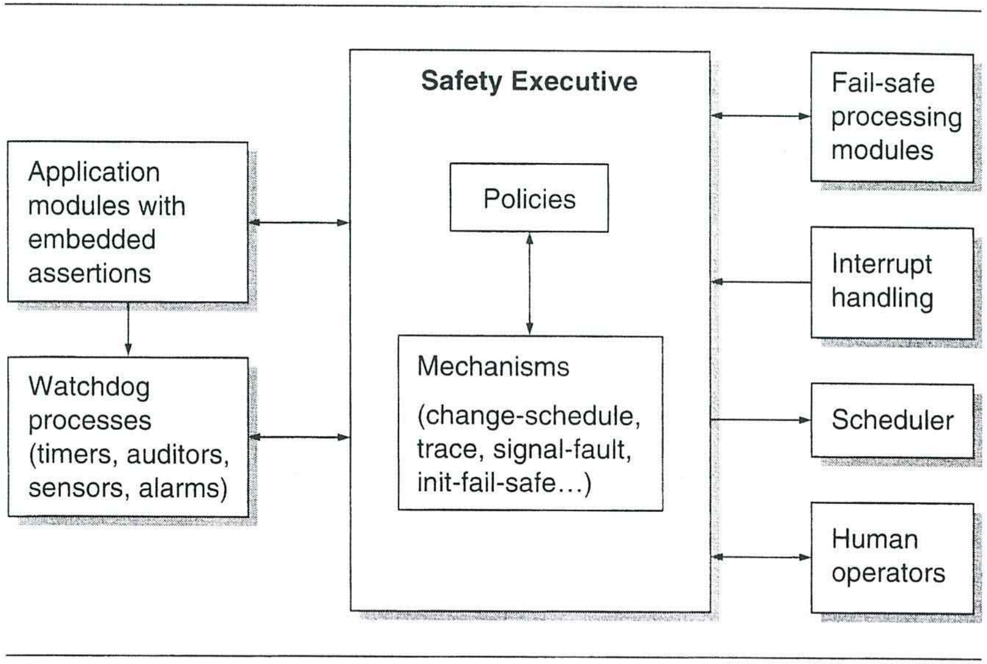

Safety Executive
Policies
Mechanisms
(change-schedule, trace, signal-fault, init-fail-safe…)
Application modules with embedded assertions
Watchdog processes (timers, auditors, sensors, alarms)
Fail-safe processing modules
Interrupt handling
Scheduler
Human operators

FIGURE 16.3
A design for a safety executive.

16.4.2 Barriers

- One way to reduce the probability of the system getting into hazardous states is to erect barriers,
  - Either physical or logical
  - Between physical objects, incompatible materials, system states, or events.

- Barriers may
  - Make access to a dangerous process or state impossible or difficult (lockout)
  - Make it difficult or impossible to leave a safe state or location (lockin)
  - Enforce a sequence of actions or events (interlock)

- Lockouts
  - Prevents a dangerous event from occurring or
  - Prevents someone or something from entering a dangerous area or state.
  - The simplest types: a wall, a fence, or some type of physical barrier

  - Useful when electrical or magnetic signals can interfere with programmable devices.
    - EMI: Electromagnetic Interference
    - EMI is a major problem for sophisticated military aircraft.
  - EMI can be eliminated or minimized in three ways [358]:
    (1) Reduced at its source
    (2) The source and the electronic device can be separated as much as possible.
    (3) A barrier can be erected around the programmable device

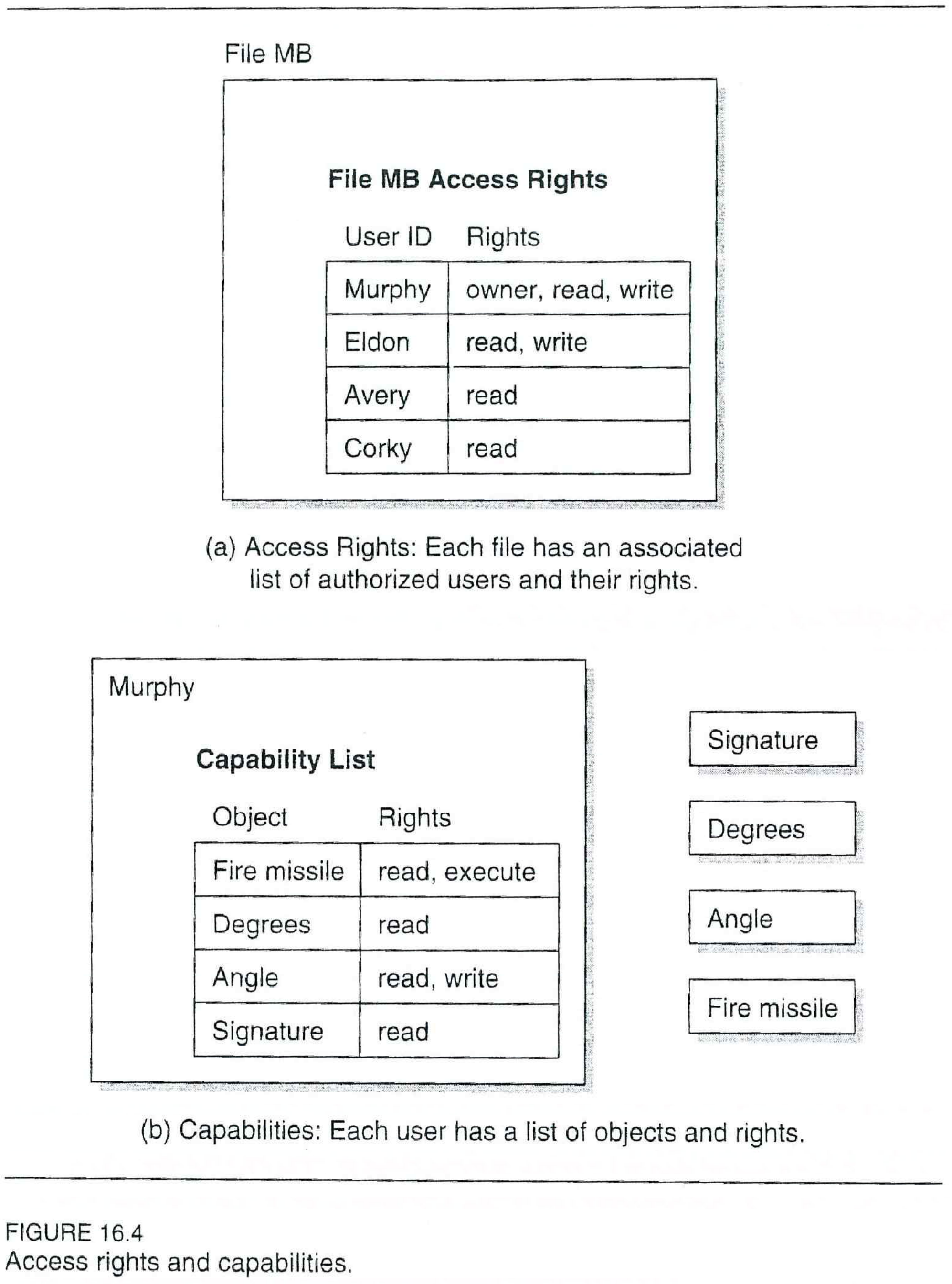
  - Authority limiting
    - A type of lockout that prevents actions that could cause the system to enter a hazardous state.
    - Inadvertent activation by human operator may be resolved by
      - By retaining a human controller in the loop and requiring a positive input by that controller before execution of hazardous commands

- Various software design technique developed to provide security may also be applicable.
  - These techniques control access by associating access rights to modules or users. (Figure 16.4a)
  - Alternatively, capability may be associated with the user of the protected component. (Figure 16.4b)

- Reference monitor
  - A more elaborate protection scheme
  - Controls all access to protected data.

- Protection subsystem
  - A solution to the more general problem of restricting communication rather than just data access
  - Performs authorization checks before allowing communication between models.
  - Security kernel

- All these lockout designs must be kept very simple or the extra protection features may only add to the software error problem.



File MB

File MB Access Rights

User ID	Rights
Murphy	owner, read, write
Eldon	read, write
Avery	read
Corky	read

(a) Access Rights: Each file has an associated list of authorized users and their rights.

Murphy

Capability List

Object	Rights
Fire missile	read, execute
Degrees	read
Angle	read, write
Signature	read

Signature
Degrees
Angle
Fire missile

(b) Capabilities: Each user has a list of objects and rights.

FIGURE 16.4
Access rights and capabilities.

- Lockins
  - Lockins maintain a condition.
  - May be used
    - To keep humans within an enclosure
    - To contain harmful products or byproducts
    - To contain potentially harmful objects
    - To maintain a controlled environment
    - To constraint a particular sequence of states or events
  - Software needs to be protected against failures and other events in its environment.
  - Software must be designed to stay in a safe state and to keep the system in a safe state despite these events.
    - Software requirements criteria to ensure that the software is robust against mistaken environmental assumptions. (Chapter 15)
    - Specifying is not enough, but the code must implement these robustness requirements.

- Interlocks
  - Commonly used to enforce correct sequencing or to isolate two events in time.
  - An interlock ensures that
    - Event A does not occur inadvertently (→ inhibits).
    - Event A does not occur while condition C exists.
    - Event A occurs before event D (→ sequencer).
  - The system should be designed so that hazardous functions will stop if the interlocks fail.
    - In addition, if an interlock brings something to halt, adequate status and alarm information must be provided to indicate which interlock was responsible [72].
  - Engineers are now often removing physical interlocks and safety features in systems and replacing them with software.
    - Introducing a more complex design
    - Hardware interlock may be important in systems with computer control in order to protect the system against software errors.
    - The software also may need to assure that
      - Proper sequences are followed and
      - Proper authority has been given to initiate hazardous actions.

- Example: Nuclear Detonation Systems

  - The goals of the system
    (1) To provide detonation when authorized
    (2) To preclude inadvertent or unauthorized detonation under normal and abnormal conditions

  - Three basic techniques
    (1) Isolation
      - Separating critical elements whose association could lead to an undesired result
    (2) Incompatibility
      - Using unique signals
    (3) Inoperability
      - Keeping the system in a state that is incapable of detonation

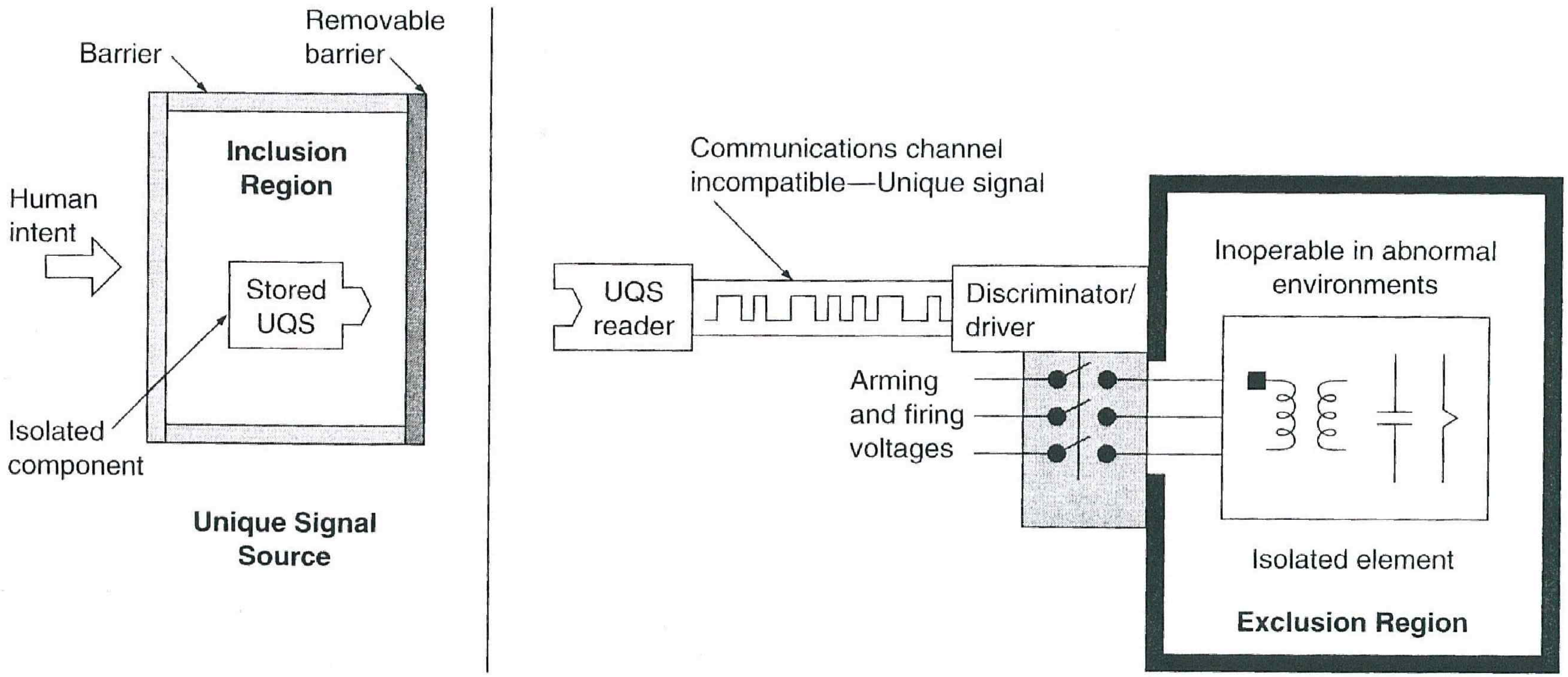


FIGURE 16.5
Subsystem using a unique signal, barriers, and inoperability for nuclear detonation safety. (Source: Stanley D. Spray. Principle-based passive safety in nuclear weapon systems, *High Consequence Operations Safety Symposium*, Sandia National Laboratories, Albuquerque, July 13, 1994.)

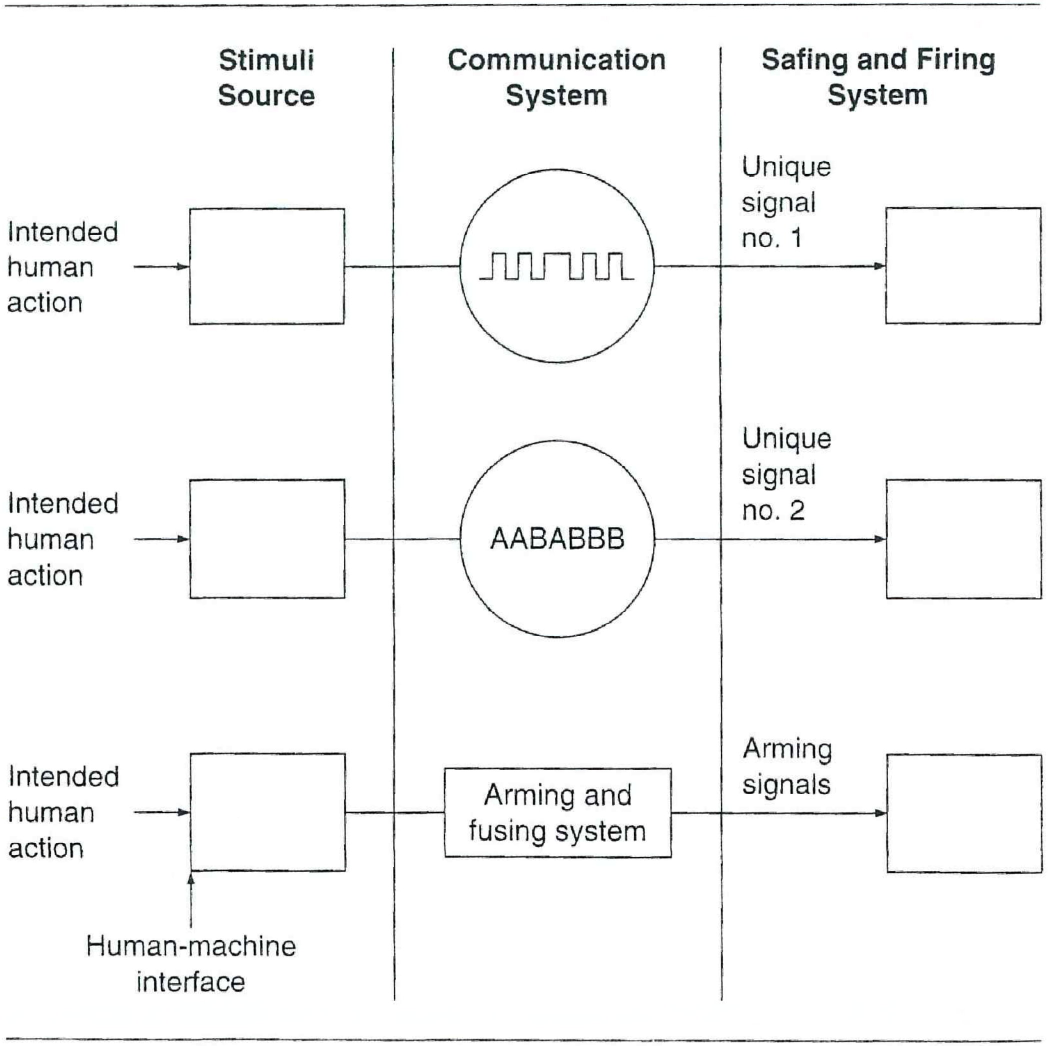
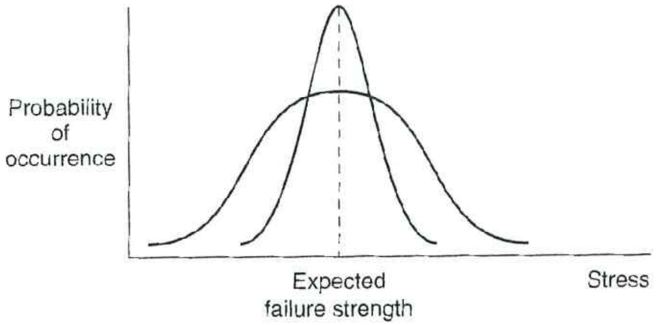


FIGURE 16.6
The use of multiple safety subsystems requiring unique signals (double direct intent with arming) to ensure proper intent. (Source: Stanley D. Spray. Principle-based passive safety in nuclear weapon systems, *High Consequence Operations Safety Symposium*, Sandia National Laboratories, Albuquerque, July 13, 1994.)

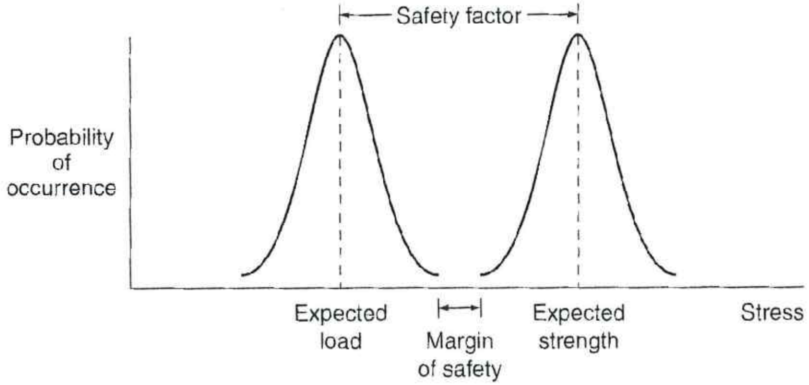
16.4.3 Failure Minimization

- Many hazards are not the result of individual component failures.
  - But, some hazard are and reducing the failure rate will reduce the probability of those hazards.
  - Section 8.5.2 briefly described several of these reliability-enhancing techniques.
  - The three most applicable to complex systems
    - Safety margins
    - Redundancy
    - Error recovery

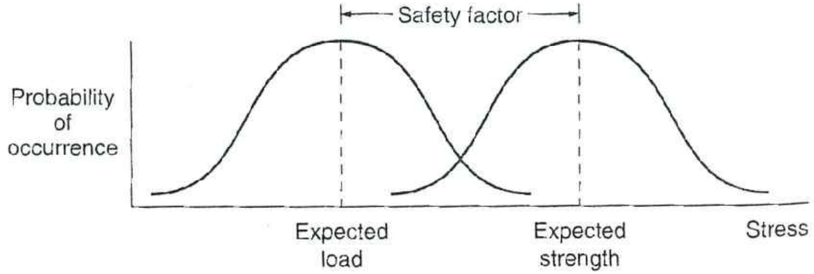
- Safety Factors and Margins
  - Engineered devices and systems have many uncertainties associated with them.
    - Engineering handbooks contain failure rates for standard components.
    - But, they may vary considerably from the mean [214].
  - Engineers have used safety factor or safety margins.
    - Involve designing a component to withstand greater stresses than are anticipated to occur. (Figure 16.7)
    - Safety factor: the ratio of nominal or expected strength to nominal stress (load)
  - Problems with safety factors
    - A calculated safety factor for a component will vary because of differences in
      - Its composition, manufacturing, assembly, handling, environment, or usage.
    - Alleviated somewhat by the use of measures other than expected value or mean in the calculations
      - Such as comparing minimum probable strength and maximum probable stress



(a) Probability density function of failure for two parts with same expected failure strength.



(b) A relatively safe case.



(c) A dangerous overlap but the safety factor is the same as in (b).

- Redundancy
  - Deliberate duplication to improve reliability.
    - Functional redundancy duplicates function, but may do it with different designs.
  - Redundancy may be achieved
    1. Through standby spare
    2. By current use of multiple devices to duplicate a function and voting on the results
       - Complex failure detection and comparison voting schemes are required.
       - Reconfiguration after switching-out may be also required.
  - Redundancy used to increase reliability will at the same time decrease safety and vice versa.
  - Functional redundancy may be accomplished
    - through identical designs (design redundancy)
    - through intentionally different designs (design diversity)
      - Diversity is used to try to avoid common-cause and common-mode failures, but providing complete diversity is difficult.
      - A vicious circle begins when redundant components introduce more complexity, which adds to the problem of common-cause failures.

- Redundancy appears to be most effective against random failures and less effective against design errors.

- Redundancy has been applied to software (for only design errors)
  - Data redundancy
  - Control redundancy

- Data redundancy
  - Includes extra data for detecting errors
  - Examples: parity bits, error-detecting and correcting codes, checksums, message sequence numbers, etc.

- Control or algorithmic redundancy
  - Has been proposed for software.
    (1) built-in reasonableness checks on the computations of the computer
    (2) Writing multiple versions of the algorithms and voting on the result

- Recovery
  - Failure can be reduced if successful recovery from the error occurs before the component or system fails.
  - Recovery can be performed by humans or can be automated. (Chapter 17)
  - Software error recovery can be forward or backward.
  - Backward recovery
    - The computer returns to a previous state and continues computations using an alternative piece of code.
    - No attempt is made to diagnose the error.
    - Multiversion software is merely a special case of backward recovery.
      - Where the versions run in parallel so that state restoration is not necessary.
    - Adequate if it can be guaranteed that an erroneous computer state will be detected and fixed before any other part of the system is affected.

- Forward recovery
  - Attempts to repair the erroneous state and processing continues without rolling back.
    - Robust data structures
      - » Use redundancy in the structure or data
      - » Ex. Linked list with backward pointers
    - Dynamically altering the flow of control
      - » Allows critical tasks to be continued while noncritical functions are delayed or temporarily eliminated.
    - Ignoring single cycle errors
  - Needed when
    - Backward recovery procedures fail.
    - Redoing the computation means that the output cannot be produced in time.
    - The software control actions depend on the incremental state of the system by a simple software checkpoint and rollback.
    - The software error is not immediately apparent and incorrect outputs have already occurred.
- In general, error-handling mechanisms, like everything else, should be as simple as possible.

16.5 Hazard Control

- Accidents can sometimes still be prevented by
  - Detecting the hazard and controlling it before damage occurs.
- Resources (both physical and information) need to be managed so that an adequate amount will be available when an emergency arises.
- Hazard control measures include
  - Limiting exposure
  - Isolation and containment
  - Protection systems
  - Fail-safe design

16.5.1 Limiting Exposure

- To stay in a safe state as long and as much as possible.
- To start out in a safe state and require a change to a higher risk state.
  - Critical flags and conditions in software should be set or checked as close as possible to the code that they protect.
  - Critical conditions should not be complementary.
    - For example, the absence of an arm condition, should not indicate the unarmed.

16.5.2 Isolation and Containment

- Barriers between the system and the environment
  - Containment vessel and shields
  - Isolate hazardous materials, operations, or equipment away from human or conditions which can cause the hazards to lead to an accident.

16.5.3 Protection Systems and Fail-Safe Design

- Moving the system from a hazardous state to a safe or safer state.
  - Fail-safe protection systems

- Hazardous states should be difficult to get into, while the procedures for switching to safe state should be simple.
  - Simple protection device: a watchdog timer
    - A timer that the system must keep restarting.
    - If the system fails to restart the timer within a given period, the watchdog initiates some type of protection actions.

  - The protection system itself should provide information about its status and control actions to the operator or bystander.
  - The designer also has to consider how to return the system to an operational state from a fail-safe state. (fallbacks)
    - Partial shutdown
    - Hold
    - Emergency shutdown
    - Manually or externally controlled
    - Restart

16.6 Damage Reduction

- Designing to reduce damage in the event of an accident

- In an emergency, there probably will be no time to assess, diagnoses, determine, and carry out that actions.
  - Emergency procedures need to be prepared and practiced.

  - "Point of no return"
    - When recovery is no longer possible or likely, and damage minimization measures should be started.

- Damage minimization techniques
  - Providing escape routes
  - Safe abandonment of products and materials
  - Devices for limiting physical damage to equipment or people

16.7 Damage Modification and Maintenance

- Design change may be necessary.

- Many accidents can be attributed to the fact that the system did not operate as intended
  - Because of changes that were not fully coordinated or fully analyzed to determine their effect on the system.

- To make design changes safely,
  - The design rationale – why particular design features were included – is needed.
  - Must be documented, updated, and compared to other accident reports.

Chapter 17

DESIGN OF THE HUMAN-MACHINE INTERFACE

Introduction

The problem, I suggest, is that the automation is at an intermediate level of intelligence, powerful enough to take over control that used to be done by people, but not powerful enough to handle all abnormalities. Moreover, its level of intelligence is insufficient to provide continual, appropriate feedback that occurs naturally among human operators. To solve this problem, the automation should either be made less intelligent or more so, but the current level is quite inappropriate … Problems result from inappropriate application, not overautomation.

- Donald Norman
The Problem with Automation

[The designers] had no intention of ignoring the human factor … But the mechanical and technological questions became so overwhelming that they commanded the most attention.

- John Fuller
-Death by Robot

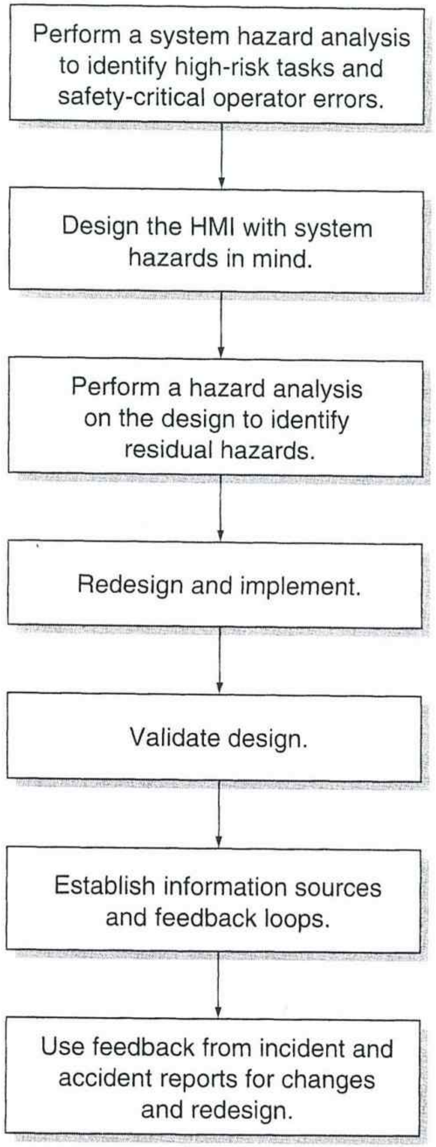
- The HMI is almost completely computer-based.
  - Operators are responsible for many more control loops than before.
  - Computers provide the flexibility to shape the HMI in ways that were once impossible.

- Now, possibilities now exist for allocating tasks and decision-making between humans and machines, with varying degrees of automation of formerly human tasks.

- We need to determine
  - How best to integrate computers into operations
  - What HMI principles and designs we should be using to enhance safety.

- This chapter outlines some traditional HMI design principles and some of the new principles that come with the introduction of computers.
  - Currently, little is known about the human-computer interface in terms of safety.

17.1 General Process Considerations



Perform a system hazard analysis to identify high-risk tasks and safety-critical operator errors.
Design the HMI with system hazards in mind.
Perform a hazard analysis on the design to identify residual hazards.
Redesign and implement.
Validate design.
Establish information sources and feedback loops.
Use feedback from incident and accident reports for changes and redesign.

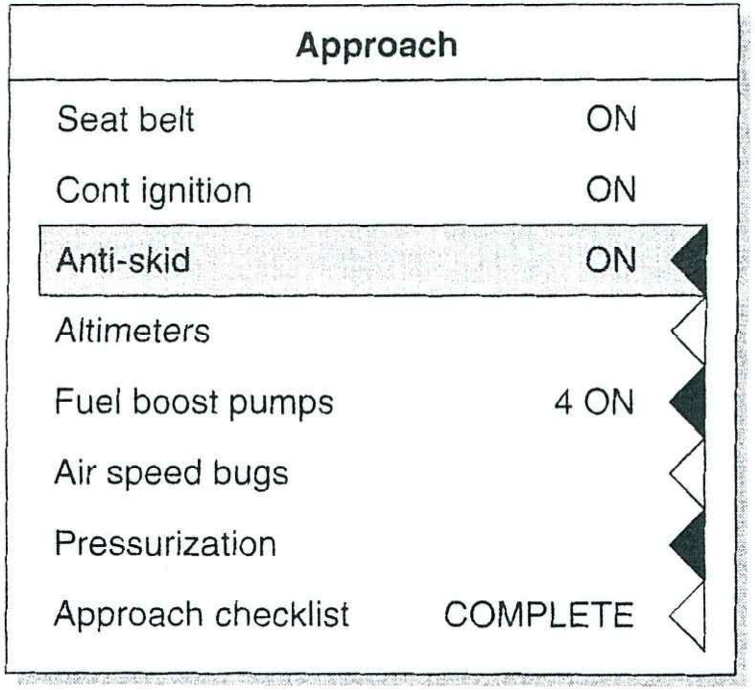
FIGURE 17.1
The process for designing a safer HMI.

17.2 Matching Tasks to Human Characteristics

17.2.1 Combating Lack of Alertness

17.2.2 Designing for Error Tolerance

- Providing Feedback about Errors
- Allowing for Recovery
- Hazard Analysis
- Unforeseen Hazards



Approach

Seat belt	ON	
Cont ignition	ON	
Anti-skid	ON	◀
Altimeters		◁
Fuel boost pumps	4 ON	◀
Air speed bugs		◁
Pressurization		◀
Approach checklist	COMPLETE	◁

FIGURE 17.2
The manual-sensed APPROACH checklist partway through execution. The first two items are colored green and the triangle symbols removed to indicate their completion. The third item has been manually acknowledged by the pilot, but the checklist software has sensed it *not* to have been accomplished. The item has been marked as skipped (amber) and the current-item-box has not advanced to the next item. The checklist will sense the state of the item only after the pilot touches the display to acknowledge completion. The pilot can either complete the item or skip it by moving the current-item-box to the next item. On touching the last item "Checklist . . . Complete," all skipped or uncompleted checklist items are displayed [251].

17.2.3 Task Allocation

- Design Considerations
- Failure Detection
- Making Allocation Decisions
- Emergency Shutdown

17.3 Reducing Safety-Critical Human Errors

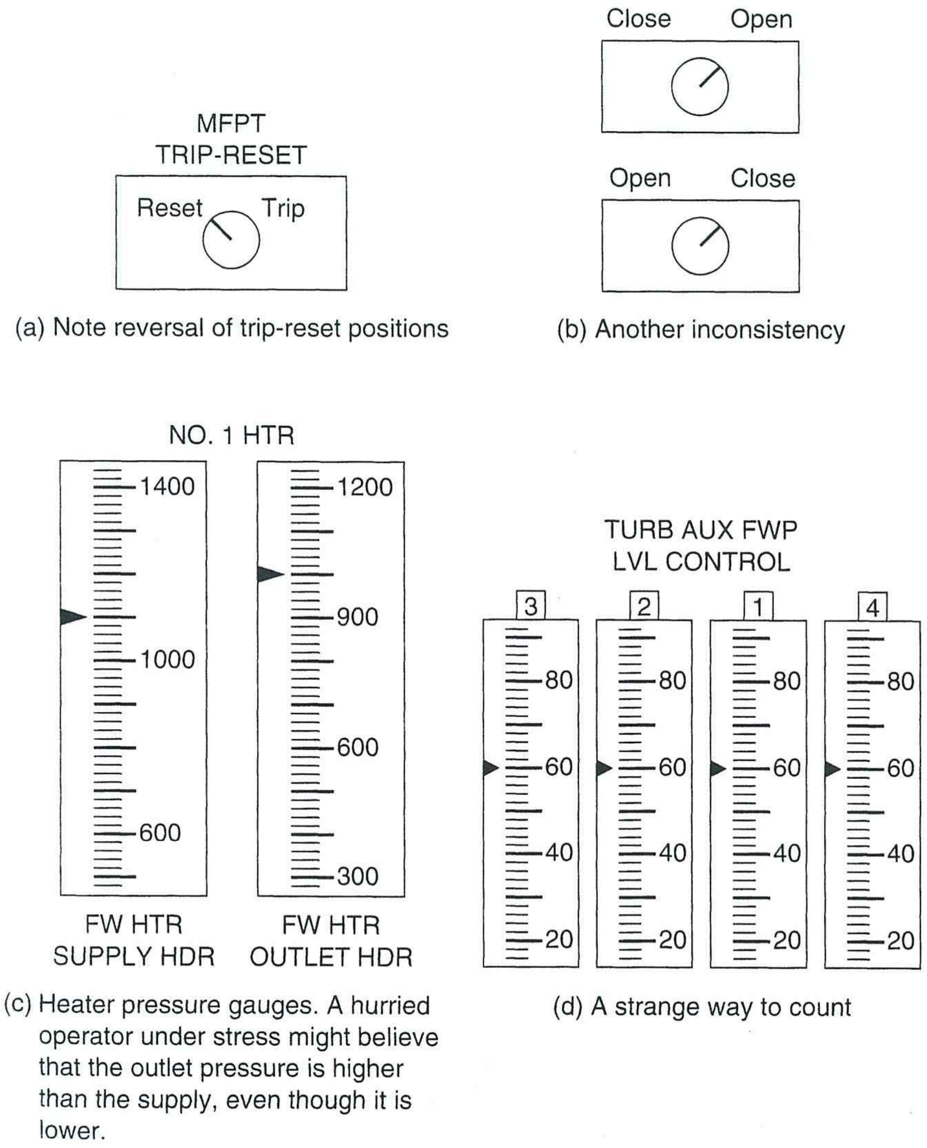
17.4 Providing Appropriate Information and Feedback

17.4.1 What Information?

- Tailoring the Display to Cognitive Processing
- Feedback
  - Feedback about the Effects of Actions
  - Feedback to Update mental Models
  - Feedback to Detect Faults
- Ease of Interpretation
- Preparing for Failure
- Alarms
- Feedforward Assistance and Decision Aids

17.4.2 Information Presentation

- Sequential vs. Parallel Presentation
- Flexibility
- Interpretation of Displayed Information
- Layouts



(a) Note reversal of trip-reset positions

(b) Another inconsistency

(c) Heater pressure gauges. A hurried operator under stress might believe that the outlet pressure is higher than the supply, even though it is lower.

(d) A strange way to count

FIGURE 17.3
Examples of poor designs seen in real control rooms.

17.5 Training and Maintaining Skills

17.5.1 Teaching Operators about Safety Features

17.5.2 Training for Emergencies

17.5.3 Simulators

17.6 Guidelines for Safe HMI Design

Chapter 18

VERIFICATION OF SAFETY

Introduction

Unlike the fairy tale Rumplestilstkin, do not think that by having named the devil that you have destroyed him. Positive verification of his demise is required.

- System Safety Handbook for the Acquisition Manager,
TU.S. Air Force

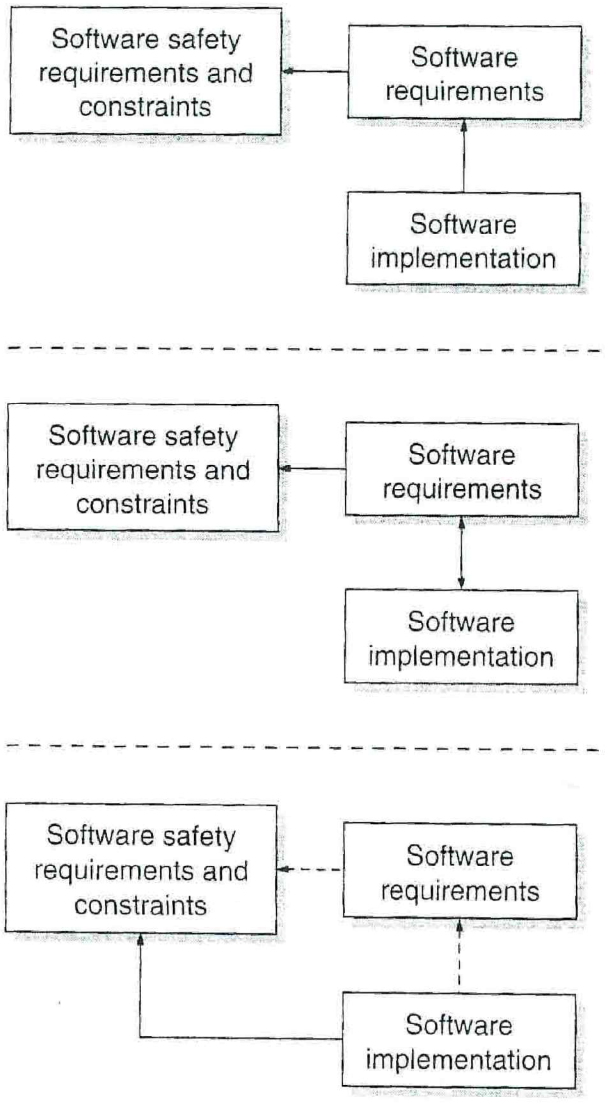
- Once the hazards have been identifies through hazard analysis and controlled through design,
  - It is still necessary to determine if any mistakes were made in these processes or in the construction of the system.

- For safety verification to be practical,
  - The software must be designed to be verifiable.

- The goal of verification is to verify that the thing being evaluated satisfies or is consistent with some criteria.
  - In the case of safety, the criteria are the system safety constraints and the safety-related functional requirements.

- Three strategies for safety verification
  - The third approach (Figure 18.1(c))
    - Guarantee that at least the identified hazards and design features to control them are considered.
    - Independent of the software requirements and design specifications, and thus able to detect errors in them.
      - Such as incomplete or omitted specification of safety-critical behavior

- Hazard analysis serves as the basis for this approach to safety verification.
  - As with hazard analysis, verification of safety is a continuous process.
  - Thus, the simple model in Figure 18.1(c) can be modified to allow verification to proceed as the software product development.


Software safety requirements and constraints
Software requirements
Software implementation
Software safety requirements and constraints
Software requirements
Software implementation
Software safety requirements and constraints
Software requirements
Software implementation

(a) Showing that the implementation satisfies the specification and that the specification satisfies the safety requirements and constraints is not adequate.

(b) This solves the unintended function problem but is impractical.

(c) Verification of the implementation directly against the safety requirements and constraints.

FIGURE 18.1
Alternative strategies.

- Two types of analysis
  - Dynamic analysis
    - Code or model of the code is executed and its performance is evaluated.
  - Static analysis
    - Code or model is examined without being executed.

- In either case, evaluating a representation of the code (specification or other type of model) is not equivalent to evaluating the code itself.
  - Due to the possibility that an abstraction or model has inadvertently omitted an important safety factor

- High assurance can only be provided by evaluating the code running on the actual computer hardware that will be used.

18.1 Dynamic Analysis

- A great deal has been written about testing.
  - This chapter focus on software and safety only.

18.1.1 Process Considerations

- Testing for safety starts with the software-related system hazards.
  - The goal is to show that the computer will not do anything hazardous from a system standpoint.

- Testing is often success oriented.
  - Focusing on doing what the software is supposed to do rather than on what it is not supposed to do.
  - A better methodology for testing rare conditions during the earlier stage could have avoided half of all failures and over two-thirds of the most critical errors [115].

- Integration testing is the first chance to verify the interface between the hardware and software.
  - Unit testing can also be involved when the safety-critical functions and constraints have been traced to individual modules and variables.

- Whenever possible, software safety testing should be integrated into the normal testing process.
  - In some special cases, safety-testing facilities may be needed or may be desired.

- Error-handling software often has errors itself.
  - Because it is less likely to be exercised during testing than is the rest of the software.

- Almost all real-time software is tested with environment simulators.
  - These and other testing tools must themselves undergo a rigorous verification process.

- Whenever changes are made to the software or to the system design,
  - Complete regression testing usually is required.
  - Safety analyses of the changes must be performed, and special unit and integration tests may be necessary.
  - If special safety testing is required, every effort should be made to integrate it into the regular testing for the change.

- In general, safety engineers
  1. Review test plans.
  2. Recommend tests based on the hazard analyses, safety standards and checklists, previous accidents and incidents, interface analyses, and so on.
  3. Specify the conditions under which the test is to be conducted.
  4. Review the test results for any safety-related problems that were missed in the analysis of in other testing.
  5. Ensure that testing feedback is integrated into the safety reviews and analyses that will be used in design modifications.

18.1.2 Limitations of Testing

- Testing for trustworthiness is an intractable task [105].
  - There are so many software paths, possible inputs, and hardware failure modes that testing for all of them is not feasible.
  - Exhaustive testing is impossible for most nontrivial software.

  - It is difficult to provide realistic test conditions.
  - Rare events that are not anticipated by the designer are equally likely to be unanticipated by the tester.
  - Using testing to assess safety is futile.

- There is a limit to the confidence that can be acquired through dynamic analysis.
  - Testing may have to be augmented with static analysis.
  - When they are used to augment each other, dynamic analysis should focus on the things that are not covered adequately by the static analysis.

18.2 Static Analysis

- Static analysis evaluates the software without executing it.
  - Instead, it examines a representation of the software.
  - More complete than dynamic analysis
    - General conclusions can be drawn and not just conclusions limited to the particular test cases that were selected.
  - Necessarily limited
    - To evaluating a representation of a behavior rather than examining the behavior itself
  - May be formal or informal.

18.2.1 Formal Verification

- Formality provides the ability to reason about, analyze, and mathematically manipulate system descriptions.

- Proving properties, especially safety properties, about the system does have the potential for increasing assurance.
  - Formal verification essentially provides a proof of consistency between two formal (mathematically rigorous) specifications of a system.

- Showing consistency between the requirements and the code is not adequate to raise confidence in safety.
  - Because most safety problems stem from flaws in the requirements.

- In general, the discipline required to make a careful, analytical argument about the software is probably helpful in finding problems and in raising confidence.

18.2.2 Software Fault Tree Analysis

- Much of the benefit of formal methods can be achieved by applying an analytical process to the problem without actually generating formal proofs.
  - Much can be gained by disciplined, focused, and careful thought about the process – exactly what is done in most of the hazard analysis techniques.
  - Unless these informal techniques are focused on safety properties, however, they are not likely to have much effect on the safety of the system.

- The system fault tree process starts out in the normal way.
  - When a leaf node describes a behavior of a computer,
  - Then Software Fault Tree Analysis (SFTA) traces that behavior into the logic of the code, and
  - Determines whether a path exists through the code that could cause the hazardous output.

- SFTA uses the same backward reasoning (weakest precondition)) approach used in formal axiomatic verification [67, 125].
  - Involves determining what must be true of the state of the computer before a statement is executed, given that the state has certain properties after the statement is executed.

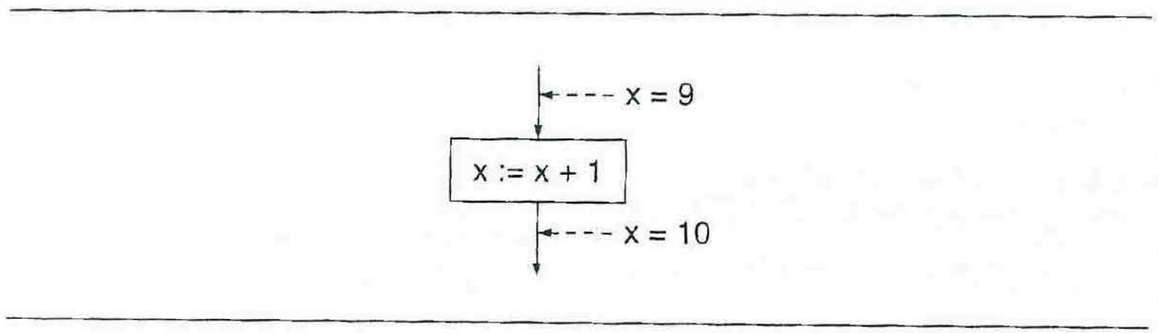


FIGURE 18.2
An example of backward reasoning about variable values.

  - SFTA follows the paths backward through the program from the hazardous outputs to identify any inputs that might cause that output.
    (It finds any legal paths through the code that produce the output.)

- The set of states or results of a program can be divided into two sets:
  - Correct
  - Incorrect

- Formal proof of correctness attempt to verify that given a precondition for the inputs and an initial computer state, the program will halt and a postcondition will be true.
  - They attempt to show that the program results in correct states.

- In safety analysis
  - The final states are divided into safe and unsafe rather than correct and incorrect.
  - SFTA attempts to verify that the program will never allow an unsafe state to be reached.
  - Specification flaws can be identified because the verification is against the system hazard analysis and not against the requirements or design specifications.

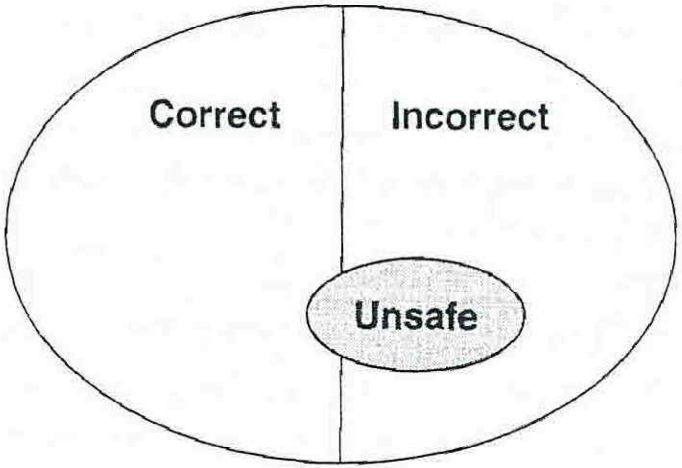


FIGURE 18.3
The relationship between correctness and safety.

- Description of the Procedure
- Uses for SFTA
- Evaluation of SFTA

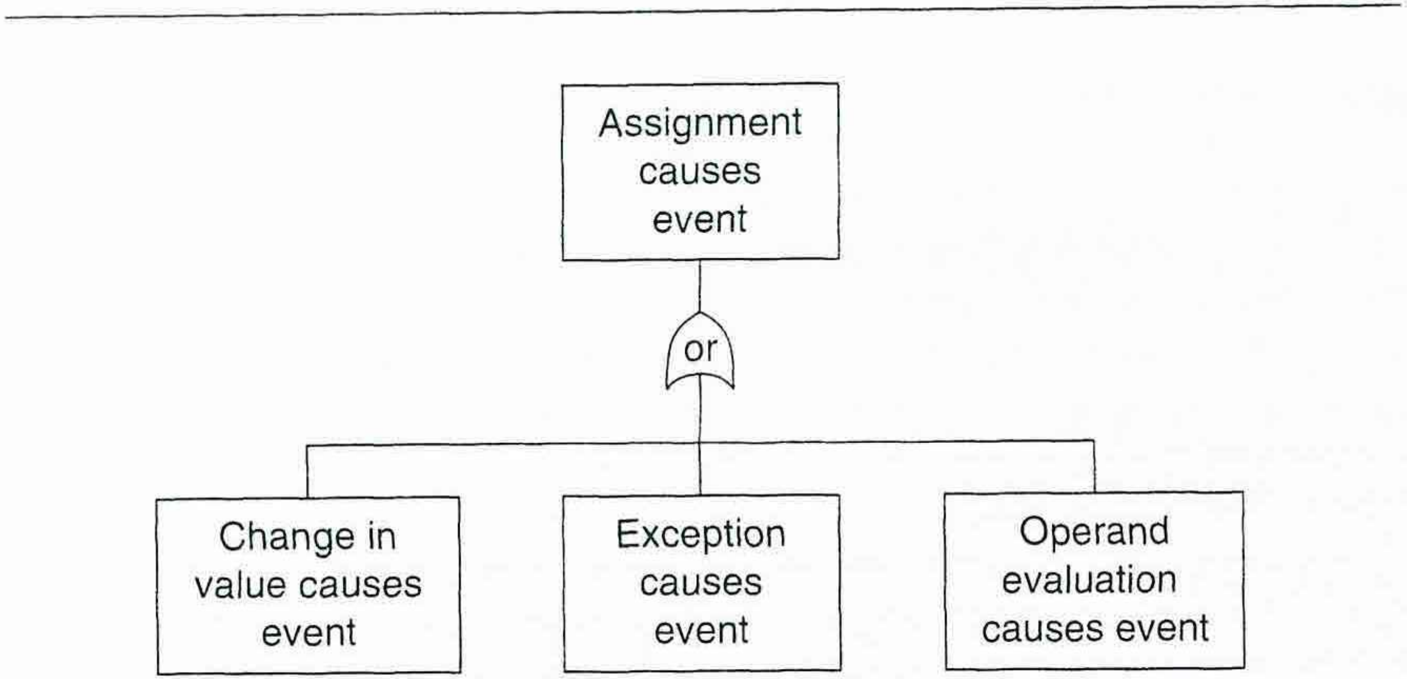
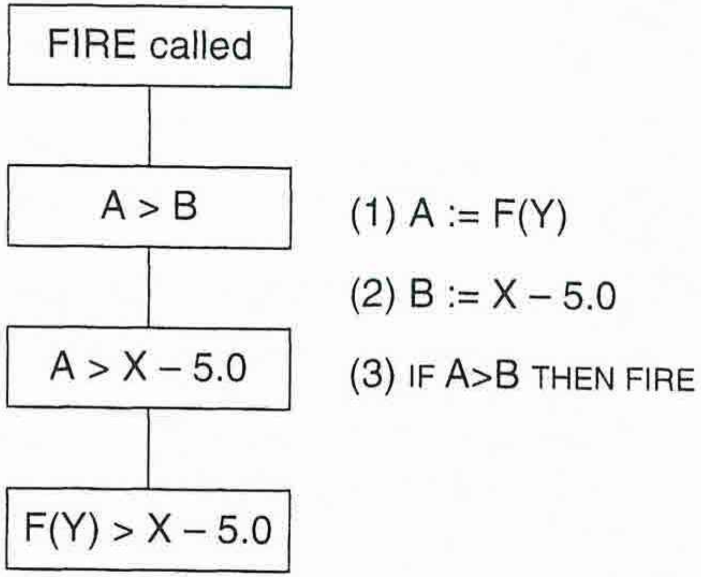


FIGURE 18.4
Template for assignment.



```
(1) A := F(Y)
(2) B := X – 5.0
(3) IF A>B THEN FIRE
```

FIGURE 18.5
Example of assignment statements and simple IF.

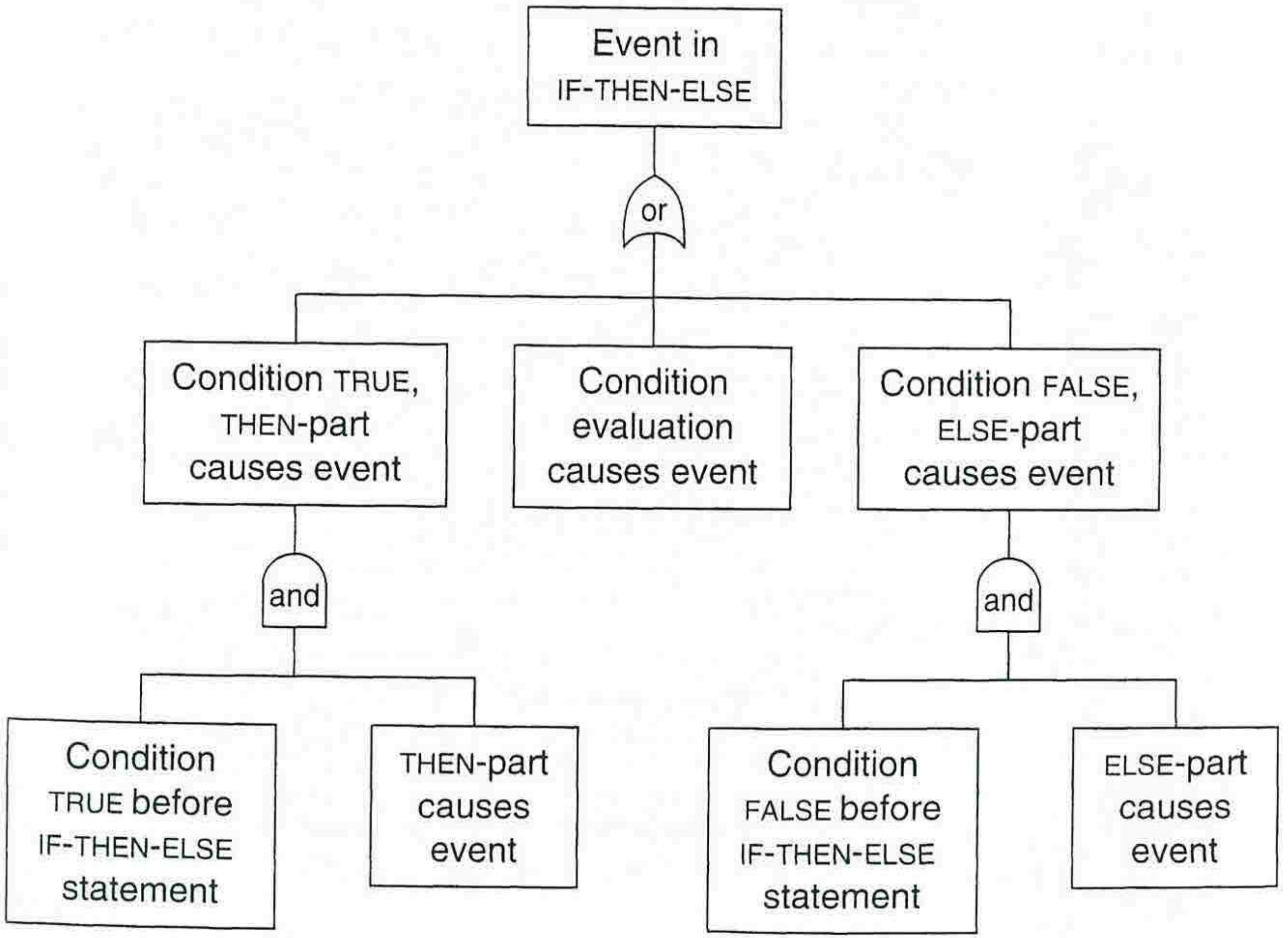

Event in IF-THEN-ELSE
or
Condition TRUE, THEN-part causes event
Condition evaluation causes event
Condition FALSE, ELSE-part causes event
and
and
Condition TRUE before IF-THEN-ELSE statement
THEN-part causes event
Condition FALSE before IF-THEN-ELSE statement
ELSE-part causes event

FIGURE 18.6
Template for IF statement.

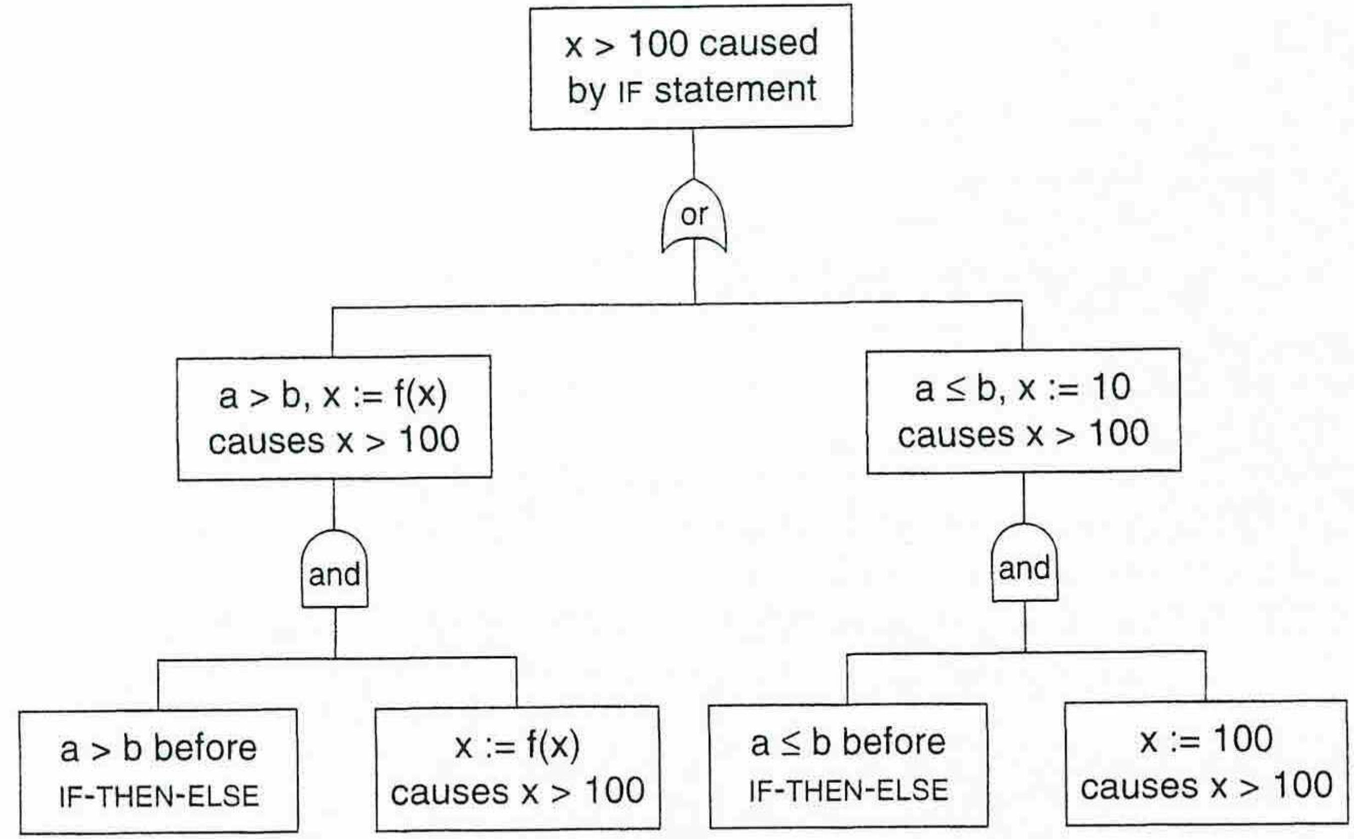

x > 100 caused by IF statement
or
a > b, x := f(x) causes x > 100
a ≤ b, x := 10 causes x > 100
and
and
a > b before IF-THEN-ELSE
x := f(x) causes x > 100
a ≤ b before IF-THEN-ELSE
x := 100 causes x > 100

FIGURE 18.7
Example for IF statement.

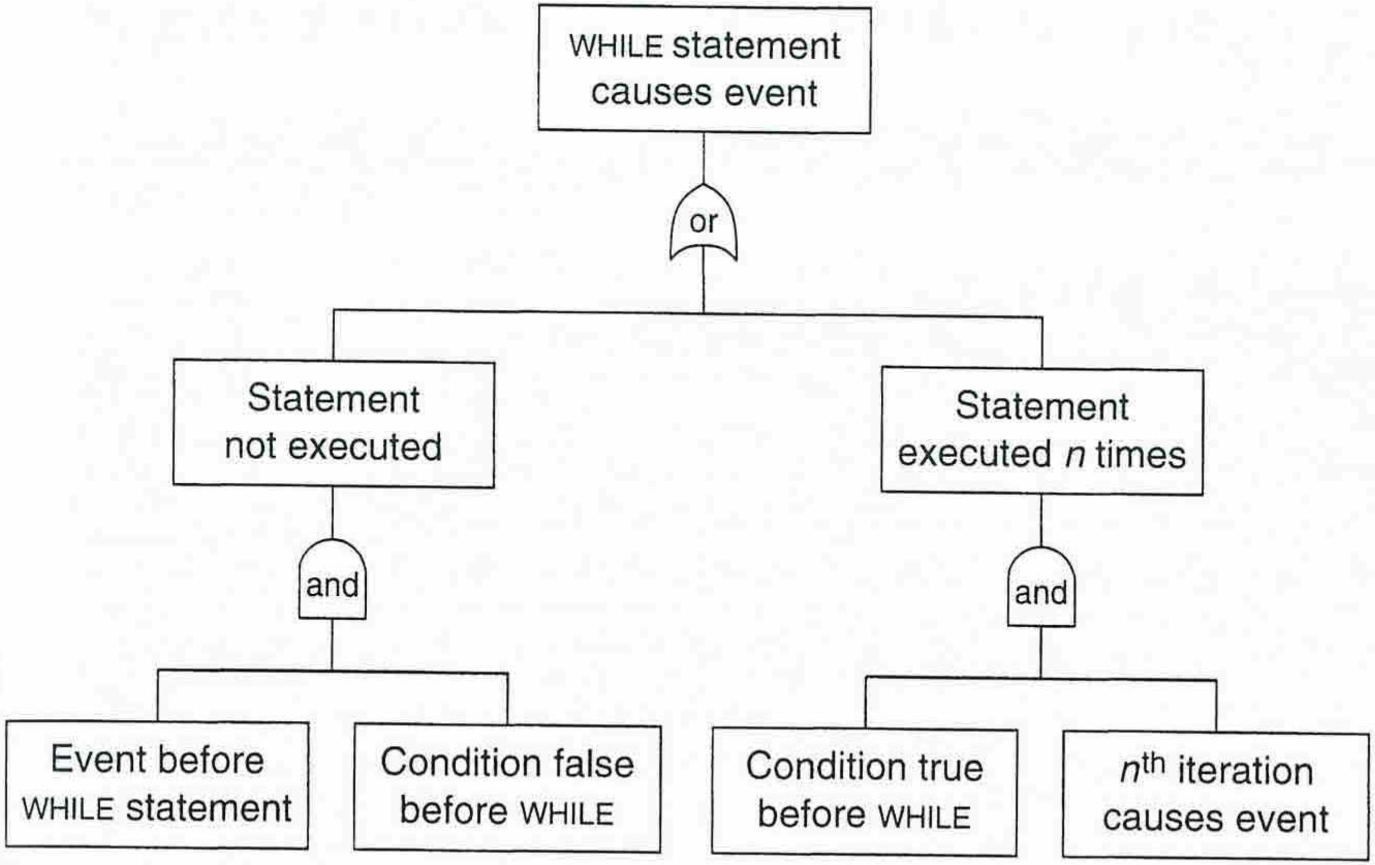

WHILE statement causes event
or
Statement not executed
Statement executed *n* times
and
and
Event before WHILE statement
Condition false before WHILE
Condition true before WHILE
*n*th iteration causes event

FIGURE 18.8
Template for WHILE statement.

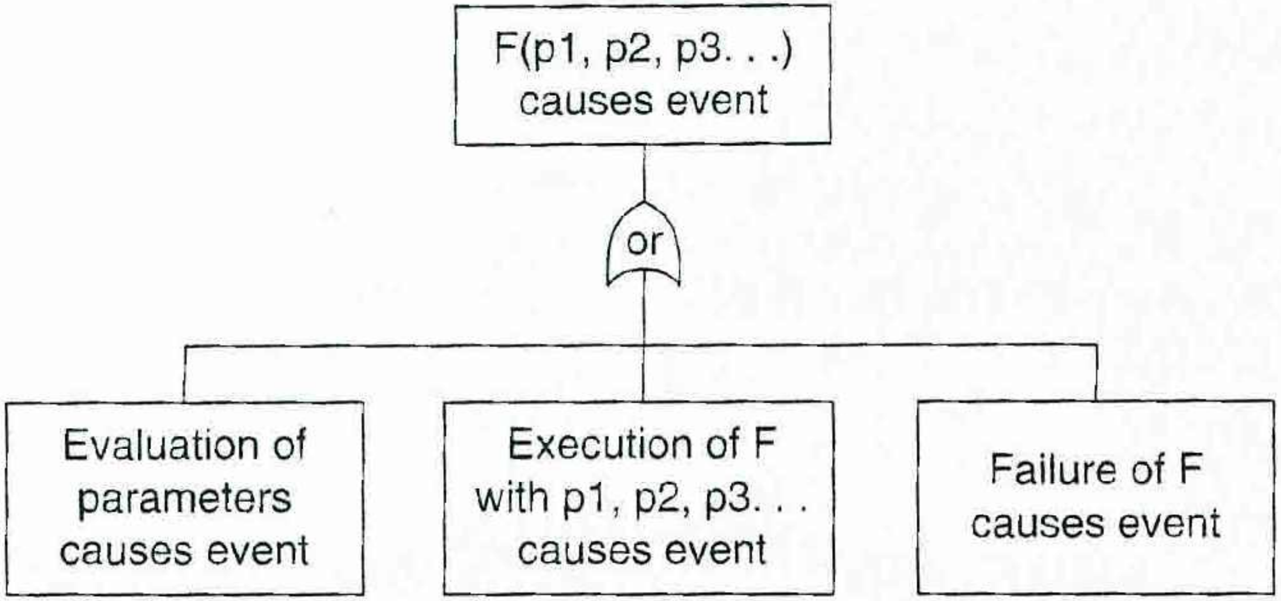


FIGURE 18.9
Template for procedure call statement.

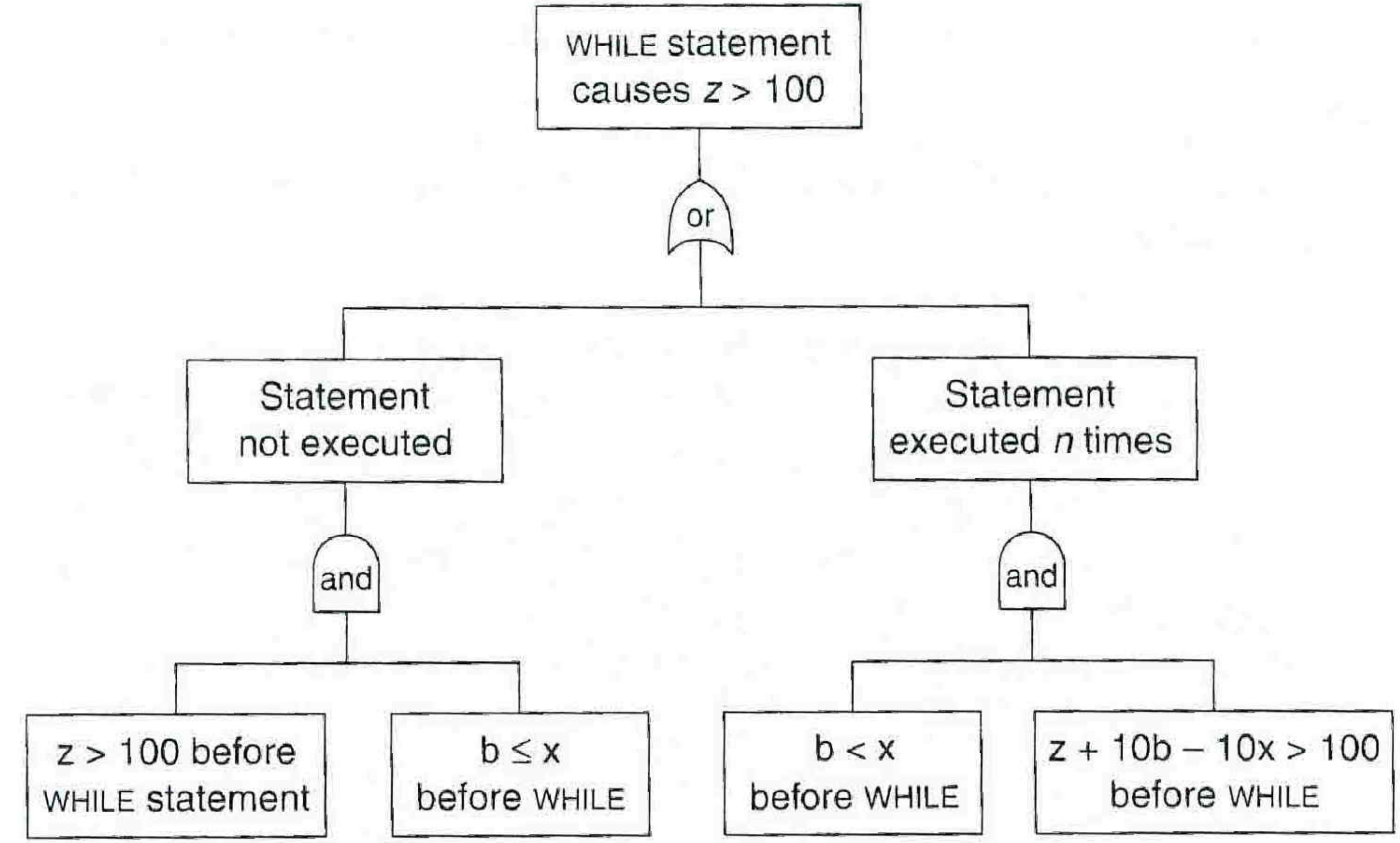

WHILE statement causes $z > 100$
or
Statement not executed
Statement executed n times
and
and
z > 100 before WHILE statement
b ≤ x before WHILE
b < x before WHILE
z + 10b – 10x > 100 before WHILE

FIGURE 18.10
Example of WHILE statement.

18.3 Independent Verification and Validation

- Independent verification and Validation (IV&V) is a process
  - in which the product of development are reviewed by an organization that is financially, technically, and managerially independent from the developer.

- The only difference from regular V&V is that it is performed by an independent organization.
  - For some special systems regulated by government agencies,
    - Nuclear Safety Cross Check Analysis (NSCCA)
    - Software Nuclear Safety Analysis (SNSA), etc.

- The effectiveness of any IV&V program is determined by
  - The specific techniques applied and the competence of those applied to them.

- Independent evaluation is limited usually
  - by being applied late in the development process and
  - by those not actually developing the software.

18.4 Conclusions

- Those all using computers for safety-critical functions are necessary but not sufficient.

- The most practical and effective way to enhance software safety is to follow a complete *Safeware* program that applies safety enhancing techniques throughout software development and maintenance.

Epilogue

THE WAY FORWARD

In theory, software development should be identical to other engineering process – we would examine known and relevant risks, and restrict our ambitions to what we knew we could handle. In practice, software invites fiendish complexity.

… There will be no breakthrough, indeed can be none … Lacking a set of inviolable natural laws to govern what software can and cannot do, we must learn to pray God. Managers and engineers must substitute their own judgment for nature’s rules. The intellectual rigor applied should be as consistent and as ruthless as Newton’s Second Law.

[Software] developers have always had to explain relationships within and between their systems. If they can explain those relationships with the simplicity and consistency demanded of other engineering disciplines, they will succeed. If not, it probably means that a dash for novelty has sprinted too far, too fast, and too soon.

- G.F. McCormick
When Reach Exceeds Grasp

- Safety is a complex, socio-technological problem for which there will be no simple solutions.
  - Our solutions will require difficult and costly procedures that in turn require special knowledge and experience on the part of system developers and maintainers.
  - Unfortunately, we have no other choice if we want to use computers to control safety-critical systems without introducing unacceptable risk.

- The basic solution is the attempts to anticipate hazards and prevent accidents before they occur.
  - It requires
    - Establishing appropriate managerial and organizational structures
    - Applying safety-enhancing techniques throughout the entire software development, maintenance, and evolution process.
    - Safeware program

- Safeware program
  - Although it cannot and will not eliminate all accidents, it is the best we can do with current knowledge.

- Certain themes run throughout the book:
  - Our most effective tool in making things safer is simplicity and building systems that are intellectually manageable.
  - Safety and reliability are different and must not be confused.
  - Placing too much reliance on probabilistic risk assessment is unwise.
  - Building safety into a system will be much more effective than adding protection devices onto a completed design. The earlier safety is considered in the development process, the better will be the results.
  - To make progress, we must stop oversimplifying accidents and recognize their complex, multifactorial nature. Fixing only symptoms while leaving root (level three) causes intact will not prevent the repetition of most accidents. Concentrating on only one or a few aspects of the problem will probably not have the desired effect. In particular, concentrating only on technical issues and ignoring managerial and organizational deficiencies will not result in effective safety programs.

- Simply replacing humans with computers will not solve the safety problem. Human "error" is integrally related to human flexibility and creativity. We should be working on ways to augment human abilities with computers rather than on ways to replace them.

- Safety is a system problem and can only be solved by experts in different disciplines working together. In particular, software cannot be developed effectively in isolation from the rest of the system. Software engineers must understand system safety concepts and techniques. At the same time, system safety personnel must be more involved in software development and include software in the system safety process.

- The safety of software can only be evaluated in the context of the system within which it operates. The safety of a piece of software cannot be evaluated by looking at the software alone.

- Complacency is perhaps the most important risk factor in a system, and a safety culture must be established that minimizes it.

- Just because the events leading to an accident are not foreseen does not mean the accident is not preventable. The hazard is usually known and often can be eliminated or reduced significantly. Often a decision is made to try to eliminate the preceding events (rather than the hazards) because that would require the fewest tradeoff with other goals, such as desired functionality or lower cost. The decisions involved are trans-scientific. But engineers have a duty to clarify the risks for decision makers and to make sure that complacency or other factors or pressures do not interfere with the engineering issues or risks being given due consideration in decision making.

- We must learn from the past so that we do not repeat the same mistakes.