A Model-based Framework for Risk Assessment in Human-Computer Controlled Systems

by

Iwao Hatanaka

Submitted to the System Design and Management Program
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Engineering and Management

Abstract

The rapid growth of computer technology and innovation has played a significant role in the rise of computer automation of human tasks in modern production systems across all industries. Although the rationale for automation has been to eliminate "human error" or to relieve humans from manual repetitive tasks, various computer-related hazards and accidents have emerged as a direct result of increased system complexity attributed to computer automation. The risk assessment techniques utilized for electromechanical systems are not suitable for today's software-intensive systems or complex human-computer controlled systems.

This thesis will propose a new systemic model-based framework for analyzing risk in safety-critical systems where both computers and humans are controlling safety-critical functions. A new systems accident model will be developed based upon modern systems theory and human cognitive processes to better characterize system accidents, the role of human operators, and the influence of software in its direct control of significant system functions. Better risk assessments will then be achievable through the application of this new framework to complex human-computer controlled systems.

Thesis Supervisor: Nancy G. Leveson
Title: Professor, Department of Aeronautics and Astronautics, MIT

Acknowledgements

I would like to thank my thesis advisor, Professor Nancy Leveson, for introducing me to the world of system and software safety and for her guidance and advice in steering this thesis in the right direction. Her pioneering and innovative methodologies in the realm of "safeware" were the inspiration and motivation for this thesis.

I am also thankful to my MIT compadres, Mario Rodriguez and Jon Demerly, for the great insights, teamwork, and fun we had on the MAPS project in Professor Leveson's safety course.

I am grateful to my parents for their support of their son's quest to seek higher education at MIT. Above all, I would like to thank my wife and soul mate, Melissa, for her continual words of encouragement, enduring patience, and understanding over the past two years while I struggled to balance family, work, and school. And two special thanks go to my daughters, Rachel and Brooke, who bring sunshine into my life and who were very supportive of their father going to school just like they were!

Table of Contents

List of Figures

Chapter 1

Thesis Introduction

1.1 Motivation

Significant advances in computer and software technology have led to the development and deployment of human-computer controlled systems at a remarkable rate. The advent of very compact, very powerful digital computers has made it possible to automate a great many processes that formerly required large, complex machinery (if they could be automated at all) [1]. However, system designs with the intention of automating human tasks have increased the complexity of the systems and have decreased the usability of the systems. Various interactions of human operators with these complex systems have led to new types of computer-related hazards and accidents. Furthermore, conventional, established methods for risk evaluation and hazard analysis are no longer applicable or effective for these new complex systems.

The motivation for this thesis is to develop a more accurate approach to risk assessment for today's complex systems and provide a means to improve the safety culture within industries and organizations. By raising awareness on the importance of system safety, the number of accidents and associated loss of human life can ultimately be reduced.

1.2 Problem Statement

Given that traditional methods are no longer effective and a new approach for risk assessment in complex systems must be developed, what fundamentals from current human error models can be integrated into a new framework? What aspects of modern systems theory and organizational theory can be incorporated into a new accident model to explain system accidents where critical operations and functions are under the control of software systems?

1.3 Goals and Objectives

This research will examine traditional risk assessment techniques in addition to current human error models and will evaluate the various strengths and weaknesses of those models. This thesis will propose a pragmatic framework consisting of a new accident model that incorporates the beneficial aspects of current human error modeling, while also accounting for the roles of software, complex cognitive processes, root causes, and traditional component failures. A case study will be drawn from the application of the new framework to a complex system called MAPS (Mobility and Positioning Software).

1.4 Thesis Outline

The chapter layout of this thesis is as follows:

- This chapter (Chapter 1) provides the introduction and motivation for this thesis.
- Chapter 2 studies the recent accidents of the Ariane 5 launcher, Titan IVB booster, Mars Climate Orbiter, and Mars Polar Lander missions.

- Chapter 3 discusses traditional risk assessment approaches.
- Chapter 4 illustrates the classic accident models utilized in accident research.
- Chapter 5 proposes a new framework for risk assessment based on a new holistic systems accident model.
- Chapter 6 presents a case study in which the new framework is applied to the MAPS system.
- Chapter 7 summarizes the research done for this thesis and makes recommendations for future research.

Chapter 2

Accidents and Causality

2.1 Overview

As defined by Leveson, an accident is "an undesired and unplanned (but not necessarily unexpected) event that results in (at least) a specified level of loss" [2]. Leveson states that although an accident is undesired or unintentional, it may or may not be a foreseen event; accidents occur even when preventive and remedial measures are planned and taken to avert an event which results in some type of damage to life, property, or the environment [2].

This chapter will briefly analyze recent accidents involving human-computer controlled systems, their accident investigations, and their true root causes for failure.

2.2 Ariane 5

The Ariane 5 launcher was a new satellite launcher jointly developed by the European Space Agency (ESA) and the Centre National d'Etudes Spatiales (CNES). As documented in the Ariane 5 Flight 501 Failure report [3], the Ariane 5 launched on its maiden flight on June 4, 1996 but veered off its flight path 40 seconds into the launch, broke up, and exploded. Key members of the Ariane 5 joint project team immediately conducted an accident investigation and constructed a chain of technical events (see Appendix A) which offered a detailed account of the failure. They concluded the primary

cause of the failure was a software exception that had occurred in both of the Inertial Reference System (SRI) units. These two duplicate SRI units were operating in parallel (the primary unit was active while the secondary unit was on stand-by in case of a malfunction in the primary) in the belief that reliability would be improved with equipment redundancy. However, when the primary SRI unit failed with the software exception, the switch-over to the secondary SRI unit could not be performed because the secondary SRI unit had also failed due to the same software exception.

The accident investigation also revealed that the software exception occurred in a portion of the software program (the alignment function) which was required only for pre-liftoff activities and did not serve any purpose once the launcher lifted-off. This function was a software requirement for the predecessor launcher, Ariane 4, but not a requirement for the Ariane 5 launcher. The report claimed "it was maintained for commonality reasons, presumably on the view that, unless proven necessary, it was not wise to make changes in software which worked well on Ariane 4" [3].

Although the Ariane 5 accident investigation specified the technical details that led to the failure in its chain of events "model", the investigation failed to adequately address the root cause of the accident. It did not sufficiently analyze the key deficiencies in management, organizational structure, social culture, and relevant policies that directly factored into the causality of the accident.

2.3 Titan IVB Centaur Failure

The Titan IVB booster was developed by Lockheed Martin Astronautics for the United States Air Force to provide heavy lift access into space equivalent to space shuttle payload capacity. With the Centaur upper stage, the Titan IVB had the capability of lifting over 13,000 pounds into geosynchronous orbit.

On April 30, 1999, the Titan IVB was launched with a Milstar military communications satellite as its payload. However, the Centaur upper stage malfunctioned, causing the stage to misfire and placed the Milstar satellite into the wrong orbit. The Air Force Space Command declared the Milstar satellite a complete loss on May 4, 1999.

The accident investigation revealed that a Lockheed Martin software engineer had entered *-0.1992476* instead of *-1.992476* in a last-minute guidance update to the Centaur software [4]. Loaded with this incorrect value, the Centaur lost all attitude control and sent it into an incorrect low orbit. Subsequently, the Milstar satellite separated from the Centaur in a fatal final orbit.

The Titan IVB accident investigation board stated that "faulty Centaur upper stage software development, testing and quality assurance process failed to detect and correct a human error made during manual entry of data values into the Centaur's flight software file" [5]. But what were the root causes that prompted the development process to fall into that state?

2.4 Mars Climate Orbiter (MCO)

On December 11, 1998, the National Aeronautics and Space Administration (NASA) launched the $125 million Mars Climate Orbiter in its mission as the first interplanetary weather satellite. The MCO was to operate for up to five years in a polar orbit to study the Martian weather and relay communications from an upcoming mission called Mars Polar Lander (MPL), which was scheduled to land on Mars in December 1999. The MCO and MPL were part of NASA's strategic program of robotic exploration of Mars to help scientists understand Mars' water history and potential for life on the planet.

Following the 9-month journey from Earth to Mars, the MCO was scheduled to fire its primary engine to commence an elliptical orbit around Mars. On September 23, 1999, NASA lost the MCO when it entered Mars' atmosphere on a trajectory lower than planned. According to the Mars Climate Orbiter Mishap Investigation Board report [6], the root cause of the MCO loss was the failure to utilize metric units in the development of a ground software file ("Small Forces") used in trajectory models. The thruster performance data was required to be in metric units according to software interface specifications; however, the data was recorded in English units. This miscommunication occurred between the MCO spacecraft team at Lockheed Martin Astronautics (LMA) in Colorado and the MCO mission navigation team at the Jet Propulsion Lab (JPL) in California.

While the MCO Mishap Investigation Board report depicted the chain of events of the accident and attempted to do some form of causal analysis (see Appendix B), the report

did not sufficiently identify the constraints or lack of constraints that allowed the contributing causes of the MCO software interface problem. What were the social dynamics and organizational culture of the Mars project teams? Did governmental or socioeconomic policies and conditions have a negative effect on the MCO project development?

2.5 Mars Polar Lander (MPL)

The Mars Polar Lander was launched on January 3, 1999 on a 90-day mission to land on Mars, study the Martian climate, and examine the subsoil for signs of water, an essential prerequisite for life. This spacecraft was designed to send its data to the MCO for relaying back to Earth; this plan was eliminated by the aforementioned loss of the MCO on September 23, 1999. However, the MPL had the ability for direct communication to Earth via its X-band radio and medium-gain antenna (MGA).

After an 11-month voyage, the MPL arrived at Mars and had a landing zone targeted near the edge of the south polar layered terrain. The lander was to be the fourth craft to touchdown on Mars and the first at the south pole. On December 3, 1999, the MPL approached Mars at its entry attitude at 12:02 p.m. PST. The lander touchdown was expected to occur at 12:14 p.m. PST, with a 45-minute data transmission to Earth scheduled to begin 12 minutes later [7]. However, the MPL failed to make contact to Earth hours after its presumed landing.

The accident investigation revealed that a spurious signal may have been generated when the landing legs were deployed at an altitude of about 1500 meters and eventually caused the MPL flight software to prematurely shut down the descent engines at 40 meters above the surface of Mars [7]. The MPL flight software was designed to initiate descent engine shutdown when the first landing leg sensed touchdown. By design, the touchdown sensors routinely generated a false momentary signal when the legs were deployed to the landing position from their stowed position. The MPL flight software was required to ignore these events; however, this requirement was not properly implemented in the software. The deployed MPL flight software incorrectly recorded spurious signals from leg deployment actions as valid touchdown events. Figure 2-1 illustrates the flawed logic in the functional flow of the MPL flight software where a software indicator was not properly reset (*Indicator State = FALSE* designated by "MISSING FROM MPL" in the Touchdown Monitor Execute function).

The logic commenced with initialization of variables in the Touchdown Monitor Start (TDM_Start) function. The logic then continued to the Touchdown Monitor Execute (TDM_Execute) function to read sensor status from the I/O card and check the Radar for 40-meter altitude (*Event Enabled = Enabled*). When the Radar detected an altitude of 40 meters, the MPL flight software started the Touchdown Monitor Enable (TDM_Enable) function. With the *Indicator State* not properly reset to *False*, the MPL flight software was designed to shutdown the descent engines given the "touchdown state" of the variables. Consequently, the MPL may have undergone a free fall due to the premature

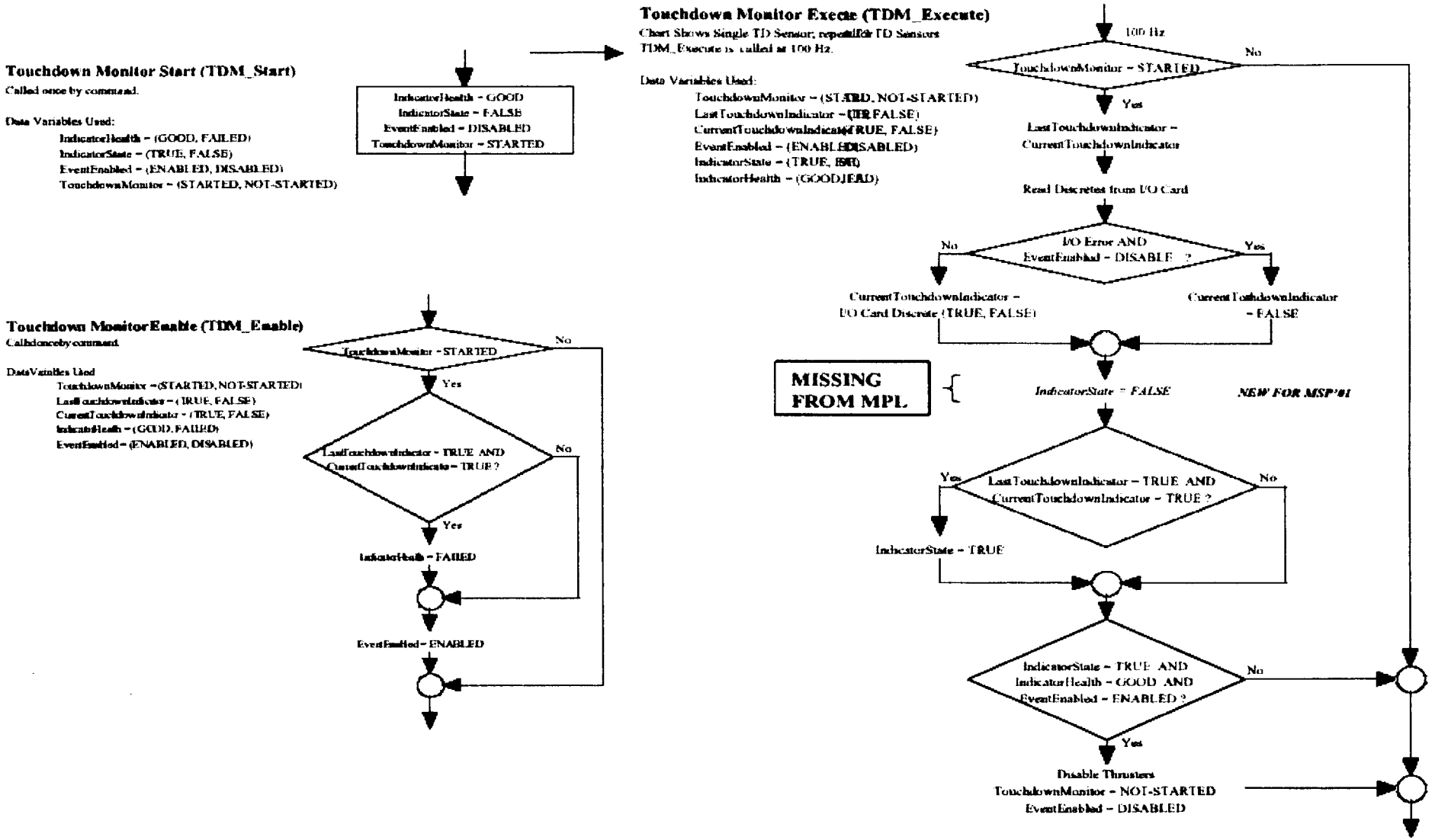


Figure 2-1: Logic Error in MPL Flight Software

engine shutdown and crashed into Mars' surface at a velocity of 22 meters per second (50 miles per hour) [7].

In the *Report on the Loss of the Mars Polar Lander and Deep Space 2 Missions* [7], the Jet Propulsion Laboratory (JPL) Special Review Board documented its findings while the MPL Mission Safety and Success Team (MSST) provided its detailed analysis on the missions. The Special Review Board investigated financial and organizational factors that may have contributed to the accident. The MPL development and operations teams were tasked with building the MPL spacecraft and landing it on Mars for approximately half the cost of the Mars Pathfinder mission. The project teams were understaffed and had to endure excessive overtime to complete the work. The tight funding constraints led to inter-group communication breakdowns and insufficient time to follow a proper development and test process.

The MPL spacecraft was on route to Mars when the Mars Climate Orbiter mission was lost for approximately 2 months. The MPL Mission Safety and Success Team (MSST) was formed to investigate any potential problems for the MPL, which may have been exposed to the same failures encountered in the MCO mission. The MSST was responsible for developing a fault-tree and failure modes analysis for the Entry, Descent, and Landing (EDL) phases of the MPL (see Figure 2-2). According to their assessment, the MSST did not have any concerns with "the description of the software design and testing provided at that time by LMA" [7]. The MSST published their findings in the JPL IOM 3130-CWW-001 report on December 1, 1999. Subsequently, the MPL mission was

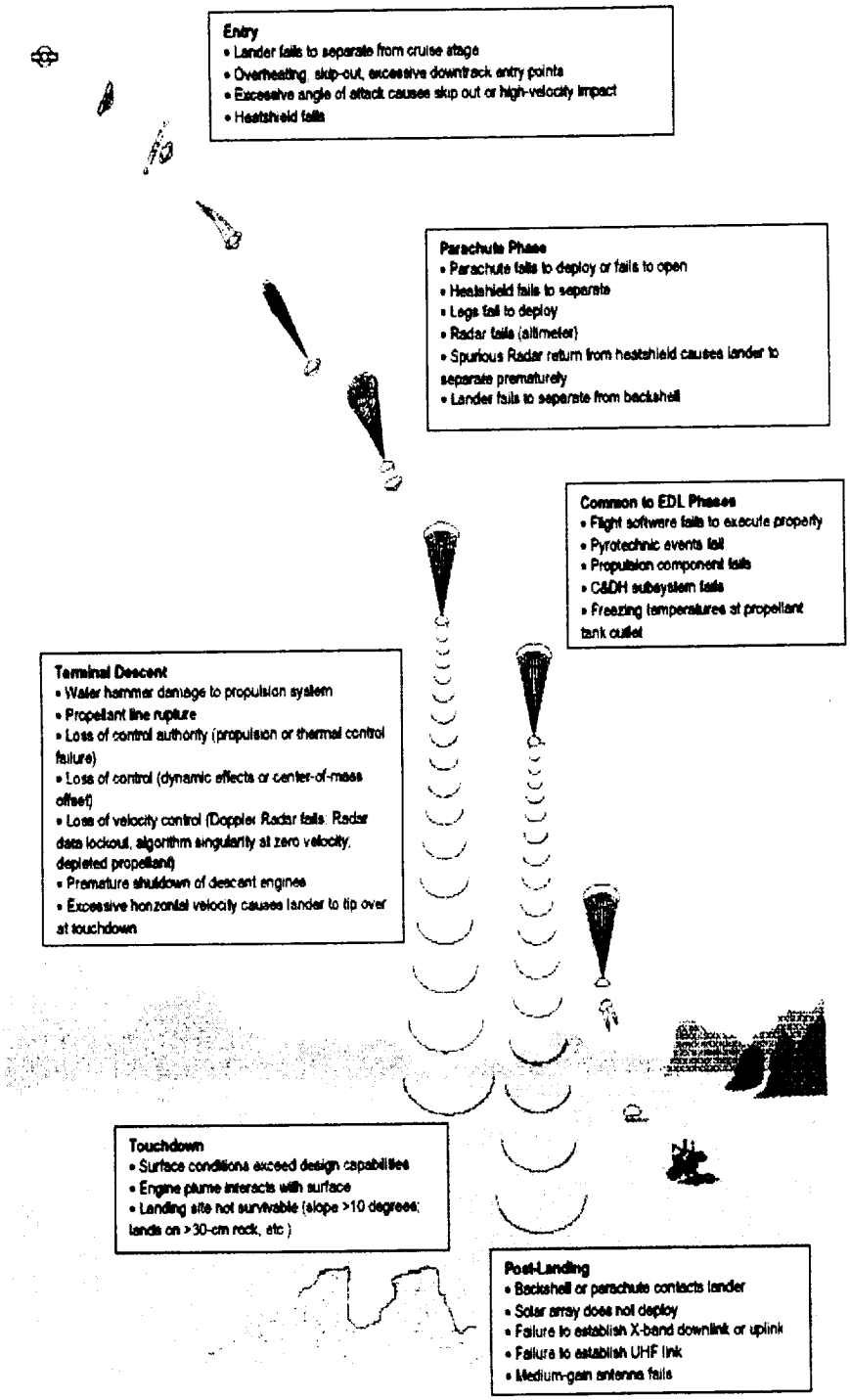


Figure 2-2: Potential Failure Mode Analysis for MPL Mission Phases

lost 2 days later. If a detailed failure mode analysis or hazard analysis had been conducted more upstream during the design/development process of the MPL, could this accident have been prevented?

The accidents and resulting investigations into causality in this chapter illustrated the various problems that have recently emerged from complex human-computer controlled systems. The next chapter will survey the strengths and weaknesses of traditional risk assessment methods utilized to assess accident risk.

Chapter 3

Traditional Risk Assessment Approaches

3.1 Overview

Leveson defines the term "risk" as "a combination of the *likelihood* of an accident and the *severity* of the potential consequences" [2]. The *likelihood* component of risk is more difficult to estimate than the *severity* component of risk. Risk assessment and analysis entails the identification and evaluation of hazards, environmental conditions, and exposure/duration. Through information obtained through proper risk assessment, the occurrence of accidents can be reduced.

Cost-benefit analysis under uncertainty, decision analysis, and probabilistic risk assessment are three traditional risk assessment methodologies pertaining to decision making that must take into consideration the presence of significant risk.

3.2 Cost-benefit Analysis Under Uncertainty

Cost-benefit analysis first gained prominence in the 1930s when the U.S. Army Corps of Engineers adopted it for evaluating water-resource projects. Its origins lie in economic theory, particularly in the economics of social welfare and resource allocation [8]. This approach aims to quantify all benefits and costs over the lifetime of a project in terms of monetary value. These streams of benefits and costs over time are compared primarily through discount rates to decide on which option has the highest proportion of benefits over risks.

Rowe offers a four-stage process for leveraging cost-benefit analysis to evaluate the relevant benefits and risks:

1) Analyze Direct Economic Benefits and Costs.
2) Analyze Indirect and Non-quantitative Effects.
3) Examine Cost of Additional Risk Reduction.
4) Reconcile Inequities.

Rowe states that "the central question in this risk-reduction analysis is determining the point at which risk has been sufficiently reduced" and acknowledges the difficulty in defining the metrics for the term "sufficiently" [8].

A weakness of cost-benefit analysis is its substantial reliance on immediate, tangible economic consequences for assessing uncertainties. Another issue is that this approach requires a monetary value be placed on the loss of human life. This primary focus on economic theory and consequences makes cost-benefit analysis an unsuitable choice for assessing and reducing risk of complex systems.

3.3 Decision Analysis

Decision analysis has its origins in the theory of individual decision making developed by von Neumann and Morgenstern (1947) and Savage (1954). This structured methodology for decision making incorporates decision trees and multi-objective analysis. A thorough decision analysis has five main steps [8]:

1. **Structure the Problem:** Identify relevant alternatives, consequences, and sources of uncertainty.

2. **Assess Probabilities:** Quantify uncertainties about present and future states of the world as probabilities.
3. **Assess Preferences:** Use subjective value judgements (i.e., utilities) and accommodate attitudes toward risk (i.e., risk aversion and risk proneness).
4. **Evaluate Alternatives:** Summarize the attractiveness of each alternative by its expected utility and weigh them by their corresponding probabilities of occurrence.
5. **Perform Sensitivity Analysis and Analyze Value of Information:** Reexamine outcomes after changing components, utilities, or probabilities and assess the value of gathering additional information that may alter the recommended decision.

The key elements of probabilities, utility functions, and structure in decision analysis are all subjective in nature. The underlying theory of decision analysis is tailored towards an individual decision-maker. Uncertainties are accommodated in the form of probabilities for calculating the utilities of options.

Probabilities in decision analysis represent an individual's degree of belief about the world, not a property of the world [8]. Decision analysis allows one to make predictions based on past failures or extrapolate from them, believing that the analyzed entity will be subject to essentially the same conditions in the future. Thus, the applicability of decision analysis is very limited for preventing accidents and reducing risk in technology-based, complex systems due to their evolving and dynamic characteristics.

3.4 Probabilistic Risk Assessment

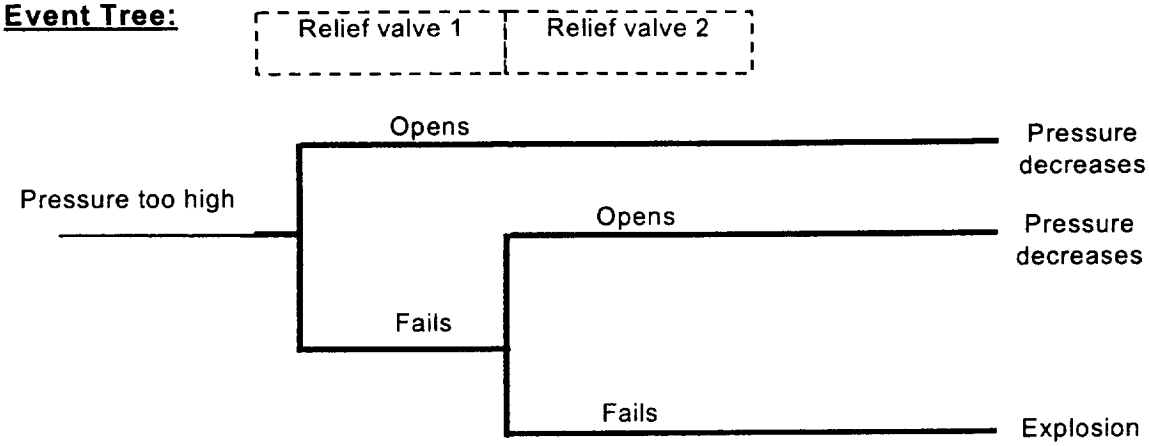
Probabilistic risk assessment (PRA) entails analyzing risk as a function of probabilities and consequences. Rowe describes risk estimation as a process that involves the following steps [9]:

- The probability of the occurrence of a hazardous event
- The probabilities of the outcomes of this event
- The probability of exposure to the outcomes
- The probabilities of 'consequences'

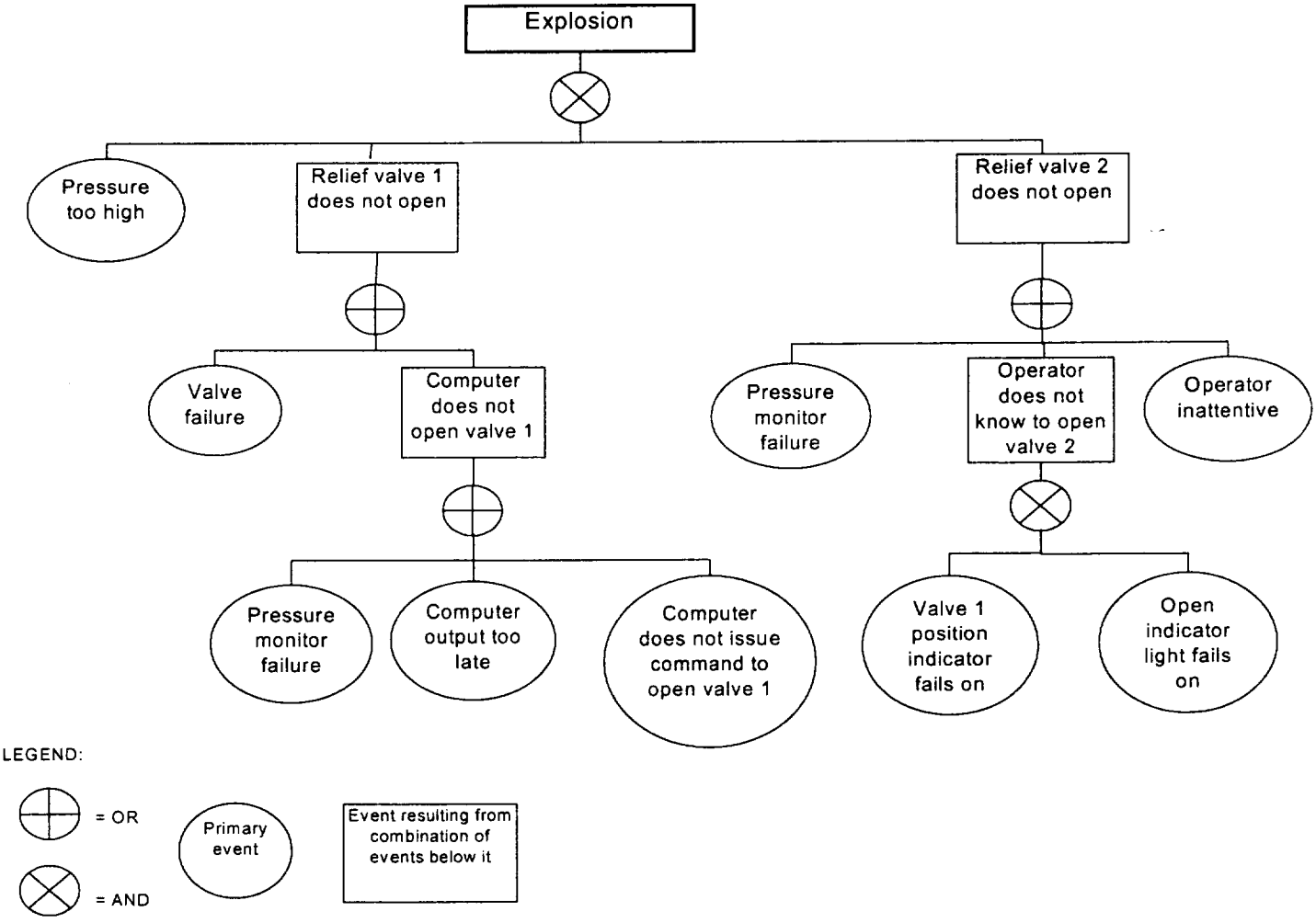
Logical tree models such as fault trees and event trees are utilized to identify prospective areas of risk and potential improvements. A fault tree starts with a system failure and then traces back to possible root causes. An event tree commences on an initiating event and progresses forward in time with consideration to failure probabilities of components between the initiating event and an unwanted result. Leveson elegantly illustrates how the same event can be displayed in an event tree and in a fault tree (see Figure 3-1).

The fault tree is a very powerful tool for investigating faults in complex systems. However, according to Lees, one limitation of a fault tree is that "it is a representation of the state of the system at one instant in time and in principle it is not well adapted to handling systems in which the defined events and states relate to different instants in time, such as a process system" [10].

Event Tree:



Fault Tree:



Source: [2]

Figure 3-1: Same Event Represented by Event Tree and Fault Tree

The event tree can be utilized to define accident scenarios as representatives of classes of events and can be associated with a finite probability value. Even so, Amendola claims that a major drawback to the event tree is that it is "a modeling technique which hardly gives an adequate account of dynamic processes interacting with systems' states, which is what an operator is in reality confronted with" [11]. Pyy and Wahlstrom also contend that "if there are possibilities for branching into more than two directions or loops and dependencies between different event trees, modeling is no longer possible by using ordinary event trees; thus, the model is not good in complex man-machine interactions" [11].

Probabilistic risk assessment requires a set of precedents to estimate over; that is, a series of developments of similar nature for which the performance and cost history is known [12]. However, for newly developed complex systems, comparable data does not necessarily exist. Leveson asserts that even if past experience is available and could be used, it might not be a valid predictor of future risk unless the system and its environment remain static, which is unlikely [13]. Complex systems of today are dynamic in nature as they attempt to keep pace with rapid advancements in technology; the system designs of these complex systems must account for new failure modes and must be developed while not knowing the entire scope of the potential risk factors and design ramifications. Small changes may substantially alter the risk involved [14]. Hence, the analysis of historical data from similar systems for estimating future risk would not be an appropriate risk assessment approach for human-computer controlled systems.

3.5 Human Reliability Analysis

Human reliability analysis (HRA) accommodates the aspects of human failures and mistakes and their effect on accident risk. The Technique for Human Error Rate Prediction (THERP) and time-reliability techniques are two HRA tools developed for use in PRA (explained in the previous section).

THERP is one of the most extensively used human reliability analysis techniques. In THERP, the operator's actions are considered in the same way as the success or failure of a system component. The operator's tasks are decomposed into task elements and essentially are component outputs in the system. The goal of THERP is "to predict human error probabilities and to evaluate degradation of a man-machine system likely to be caused by human errors alone or in connection with equipment functioning, operational procedures and practices, or other system and human characteristics that influence system behavior" [15]. THERP utilizes a form of the event tree called the "probability tree diagram." The four-step process for THERP is as follows:

- Identify the system functions that may be influenced by human error.
- List and analyze the related human operations.
- Estimate the relevant error probabilities based on historical data and expert judgement.
- Estimate the effects of human error on system failure events.

Time-reliability techniques such as operator action trees (OATS) focus on quantifying post-accident errors on the basis of time-reliability curves. The OATS technique assesses

operator errors during accident and aberrant conditions and provides error types with associated probabilities to be used in conjunction with PRA. Reasoning, diagnosis, and strategy selection are the types of cognitive errors that OATS concentrates on. The cognitive errors are quantified by time-reliability curves, which identify the probability of failure as a function of the time interval between the moment at which the relevant warning signals are evident to when action should be taken to achieve successful recovery [16].

While these human reliability assessment approaches provide powerful tools to assess risk, the estimation of the probabilities for human error in complex systems is a very difficult facet intrinsic to these approaches. According to Rasmussen, human variability in cognitive tasks, slips of memory, and mistakes are difficult to identify or to use for predictive purposes. Rasmussen claims that "the sequence of arguments an operator will use during problem solving cannot be described in general terms, the goal to pursue must be explicitly considered, and the actual choice depends on very subjective and situation-dependent features" [17]. Thus, Rasmussen's reasoning implies that operator tasks cannot be separated from their context.

Another weakness of human reliability assessment is that the techniques do not apply to emergency situations; Leveson has found that very little data on human errors in emergencies is available [2]. In general, the probability of ineffective behavior during emergency situations is greater than during normal processing; the probability of error decreases as response time increases. Hence, probability rates for human error are

difficult to derive from accident reports since cognitive data with respect to emergency situations is not easily attainable.

Given these traditional risk assessment techniques, the next chapter will focus on traditional accident models that explain how accidents have occurred.

Chapter 4

Traditional Accident Models

4.1 Overview

For those accidents that do occur, accident models provide a mechanism for understanding the events and conditions that led to the resulting accident. Accident models also offer a means to learn how to prevent future accidents from happening. This chapter will present a survey of key accident models.

4.2 Process Models

A process model for an accident represents an accident as a flow of events with time as the basic variable. The domino theory model, multilinear events sequence model, and chain-of-events model are examples of process models.

4.2.1 Domino Theory (Heinrich)

Developed by Heinrich in 1959, the domino theory of accidents is an early variant that models an accident as a one-dimensional sequence of events [17]. The five factors in the domino theory are:

1. Ancestry and social environment
2. Fault of person
3. Unsafe act and/or mechanical or physical hazard
4. Accident
5. Injury

One step is dependent on another and one step follows because of another. Thus, the model comprises a sequence that may be compared with a row of dominoes placed on end and in such alignment in relation to one another that the fall of the first domino precipitates the fall of the entire row [18]. Heinrich suggests that the removal of the central domino (unsafe act or hazard) will lead to accident prevention.

4.2.2 Updated Domino Theory (Bird)

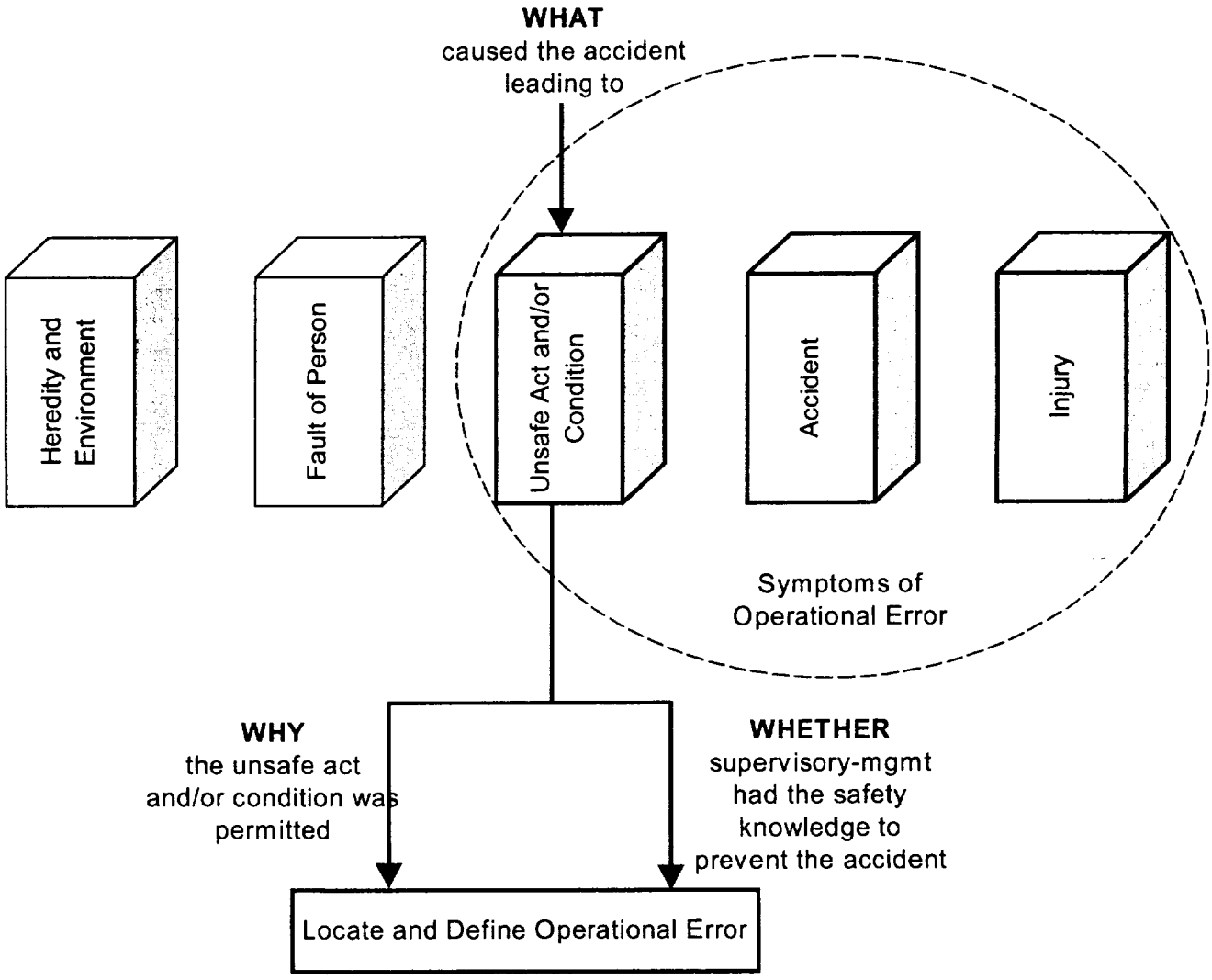
In 1974, Bird updated the domino theory to incorporate the following five key loss control factors in an accident sequence [18]:

1. Lack of control - Management
2. Basic cause - Origins
3. Immediate causes - Symptoms
4. Accident - Contact
5. Injury - Damage, Loss

Bird's theory introduces the need to assess the impact of management systems and managerial error in the causation sequence.

4.2.3 Updated Domino Theory (Weaver)

Weaver made another update to the domino theory by strongly focusing on operational error. As depicted in Figure 4-1, Weaver developed the notion of "locate and define operational error" in order to facilitate causal analysis and corrective action for supervisory-management practices.



Source: [18]

Figure 4-1: Updated Domino Theory (Weaver)

Weaver claims that "behind any proximate cause (unsafe act and/or condition) ascribed to an accident lie management practices in policy, priorities, organization structure, decision-making, evaluation, control, and administration" [18]. The "whether" question pertains to whether or not the organization possessed knowledge of the available safety

technology. The "why" question scrutinizes the operational errors in areas such as management policy, responsibility, and rules.

4.2.4 Multilinear Events Sequence Model

Benner created the multilinear events sequence model in 1975 to model the process of an accident as a succession of events that represent the interactions between various actors of the system. Also called the P (for perturbation) theory of accidents, this model depicts an injury as the result of an actor failing to adapt to disturbances within the system.

4.2.5 Chain-of-Events Accident Model

The chain-of-events accident model organizes causal factors for accidents into a chain in chronological sequence. By eliminating or modifying specific events in the chain, the accident may be prevented.

As an example, the authors of the Ariane 5 Accident Report followed the chain-of-events accident model through explaining the chronological sequence and relationships of technical events that led to the Ariane 5 failure. In Section 2.1 *"Chain of Technical Events"*, the report documents "the chain of events, their inter-relations, and causes" by "starting with the destruction of the launcher and tracking back in time towards the primary cause" [3]. The report also concludes that "it is established beyond reasonable doubt that the chain of events set out above reflects the technical causes of the failure of Ariane 501" [3]. A chain-of-events accident model for the Ariane 5 could look like the one below:

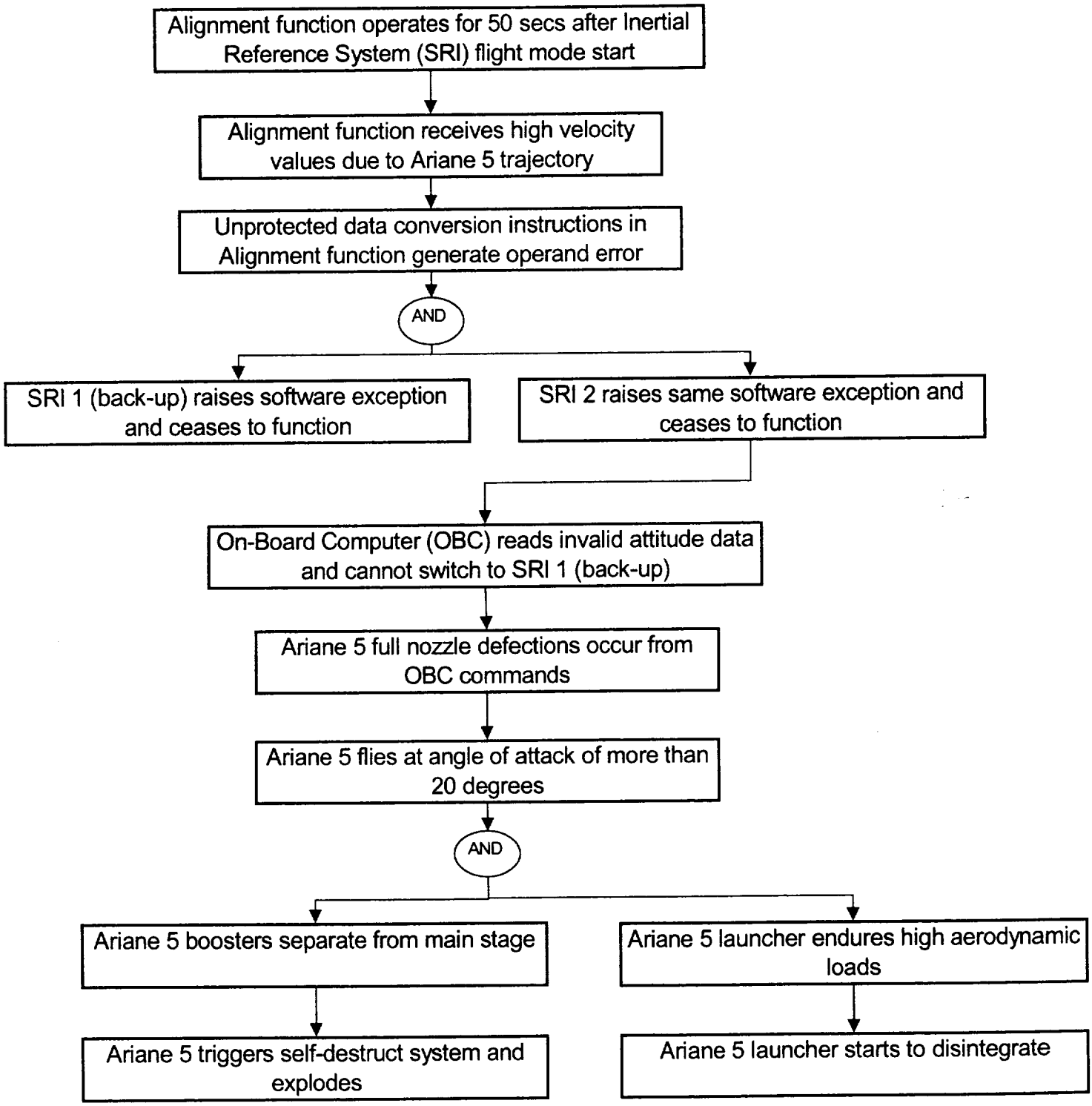


Figure 4-2: Proposed Chain-of-Events Accident Model for Ariane 5 Failure

A weakness that Leveson points out in the chain-of-events accident model is that there is no real stopping point when tracing events back from an accident, yet many of the preceding events are not relevant to the design of prevention procedures [2]. Another drawback is the subjectivity in the choice of which events are included in a chain-of-events accident model.

4.2.6 INRS Model

The INRS (French National Institute of Scientific Research) model by Monteau in 1977 is a chain-of-events model illustrating that error production results from changes in the "usual" condition [18]. These changes or deviations are assigned to one of the following categories: the operator, the machine, the surrounding environment, and the man-machine interaction (task).

The INRS model organizes events as event chain relationships or confluence relationships. The event chain relationship signifies that event Y occurs if event X occurs; this relationship is denoted by $X \rightarrow Y$. The confluence relationship event Y occurs if independent events X_1 and X_2 occur; this relationship is symbolized by:

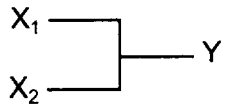
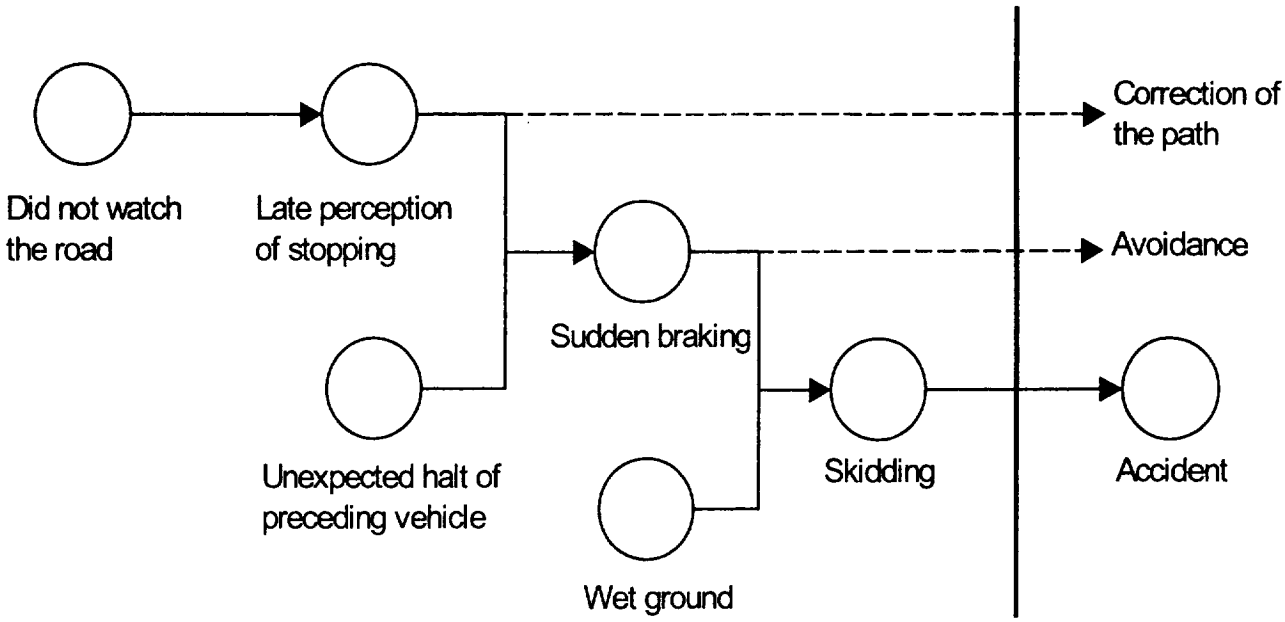


Figure 4-3 shows an analysis of event relationships in an accident scenario using the INRS model.

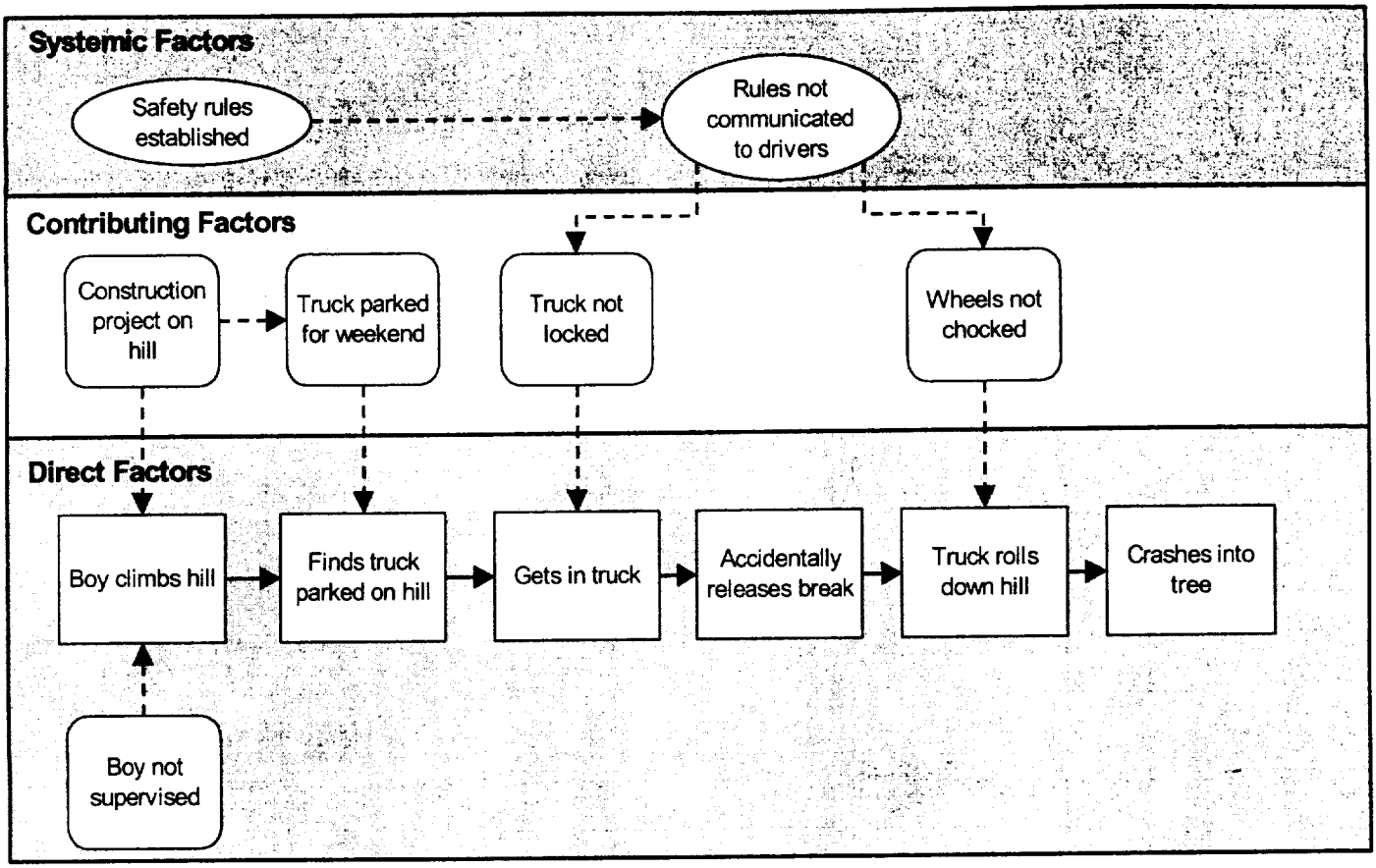


Source: [18]

Figure 4-2: INRS Model

4.2.7 NTSB Model

The preceding chain-of-events models did not adequately facilitate the incorporation of societal conditions, organizational structures, safety culture, or other system-related factors. The National Transportation Safety Board (NTSB) model of accidents was developed in the 1970s to provide a model and a sequencing method that described accidents as patterns of direct events/factors stemming from contributory factors, which in turn arise from systemic factors [2]. Figure 4-3 illustrates an example of an NTSB model for an accident.



Source: [2]

Figure 4-3: NTSB Model

4.2.8 MORT Model

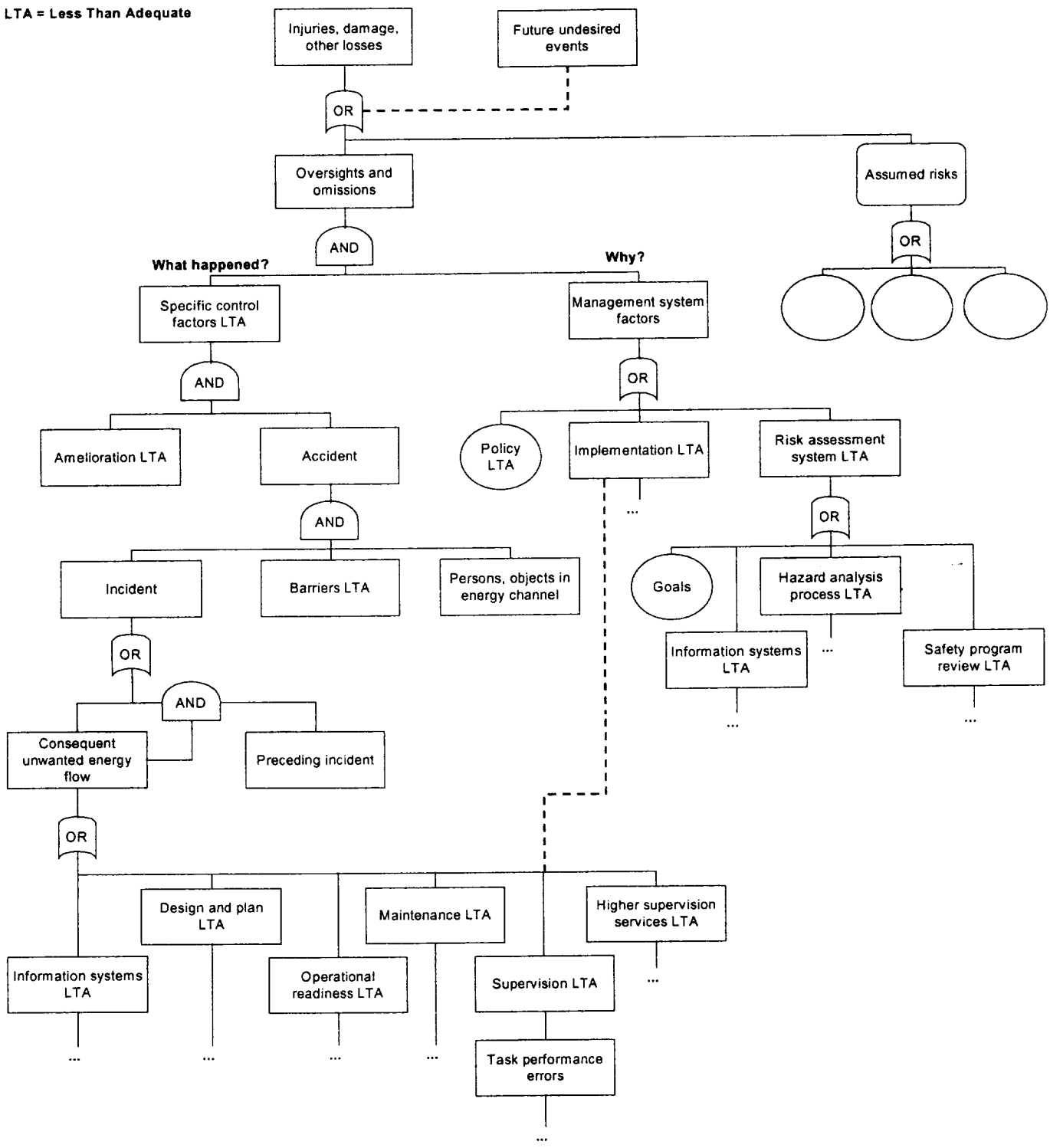
Johnson argues that simple chain-of-events models fail to recognize the role of purpose, goal, performance, and supervisory control and intervention, and thus are not appropriate for occupational accidents [2]. Johnson created the Management Oversight and Risk Tree (MORT) model to accommodate multiple system factors and conditions with respect to management, safety programs, supervision, and maintenance, among others.

Johnson based his model on the following six error-reduction concepts [19]:

1. Errors are inevitable (rate-measurable) concommitant [*sic*] of doing work or anything.
2. Situations may be error-provocative – changing the situation will likely do more than elocution or discipline.
3. Many error definitions are "forensic" (which is dabatable [*sic*], imprecise, and ineffective) rather than precise.
4. Errors at one level mirror service deficiencies at a higher level.
5. People mirror their bosses – if management problems are solved intuitively, or if chance is relied on for non-accident records, long-term success is unlikely.
6. Conventional methods of documenting organizational procedures seem to be somewhat error-provocative.

Johnson proposes that human errors existing at lower levels of the organization are symptoms of problems at higher levels of the organization. He also suggests that human error can be reduced through a change in the situation. This change can be accomplished by assistance from the outside (staff safety, line management, etc.), working within a corporate philosophy, through study of the situation, and through participation of the individual worker [19].

Johnson based the format of the MORT model on logic diagrams and checklists; accident factors and evaluation criteria are presented in a fault tree format and are connected by logical gates to show relationships. Johnson's MORT model in Figure 4-4 shows how injuries, damage, or other losses arise from job oversights or assumed risks.



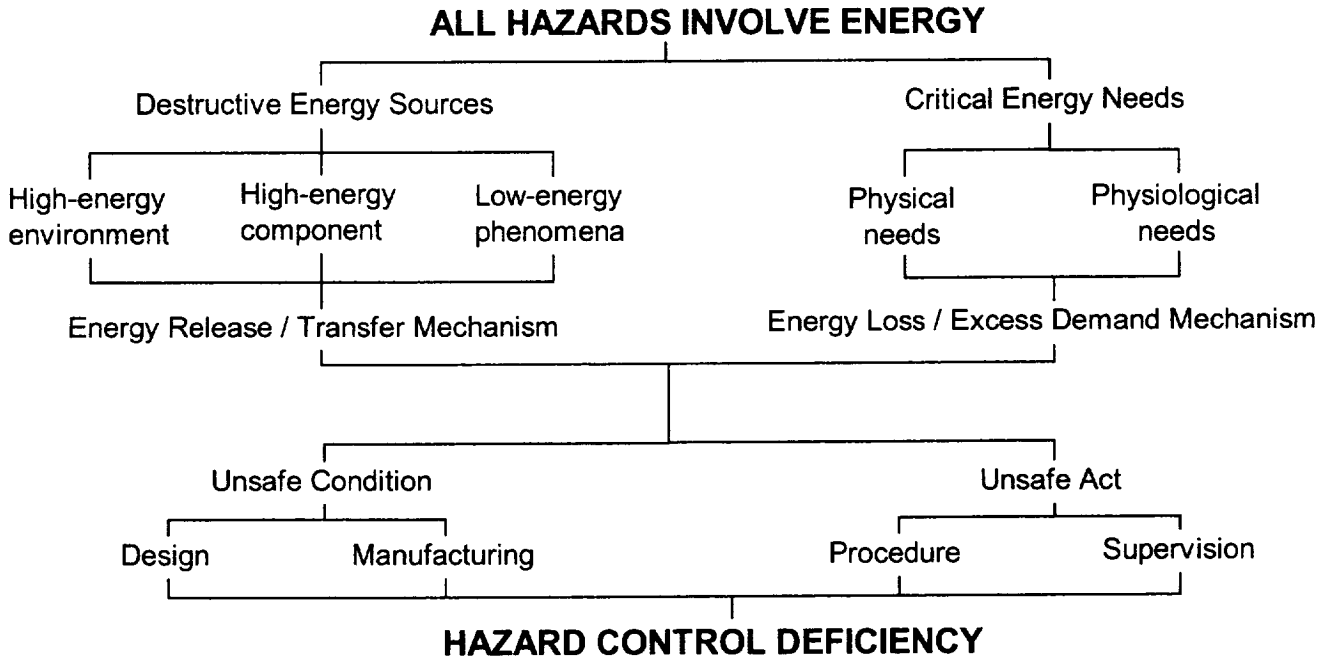
Source: [2]

Figure 4-4: MORT Model

4.3 Energy Models

As one of the earliest models to explain accident causality, the energy model characterizes accidents as the outcome of an unwanted or uncontrolled release of energy. According to this model, accidents can be prevented by establishing barriers between the energy source and the object that may be affected.

Ball developed a causation model based on the concept that energy release is a primary factor in the cause of accidents. The Ball energy model (see Figure 4-5) suggests that all accidents are caused by hazards, and all hazards involve energy, either due to involvement with destructive energy sources or due to a lack of critical energy needs [18].



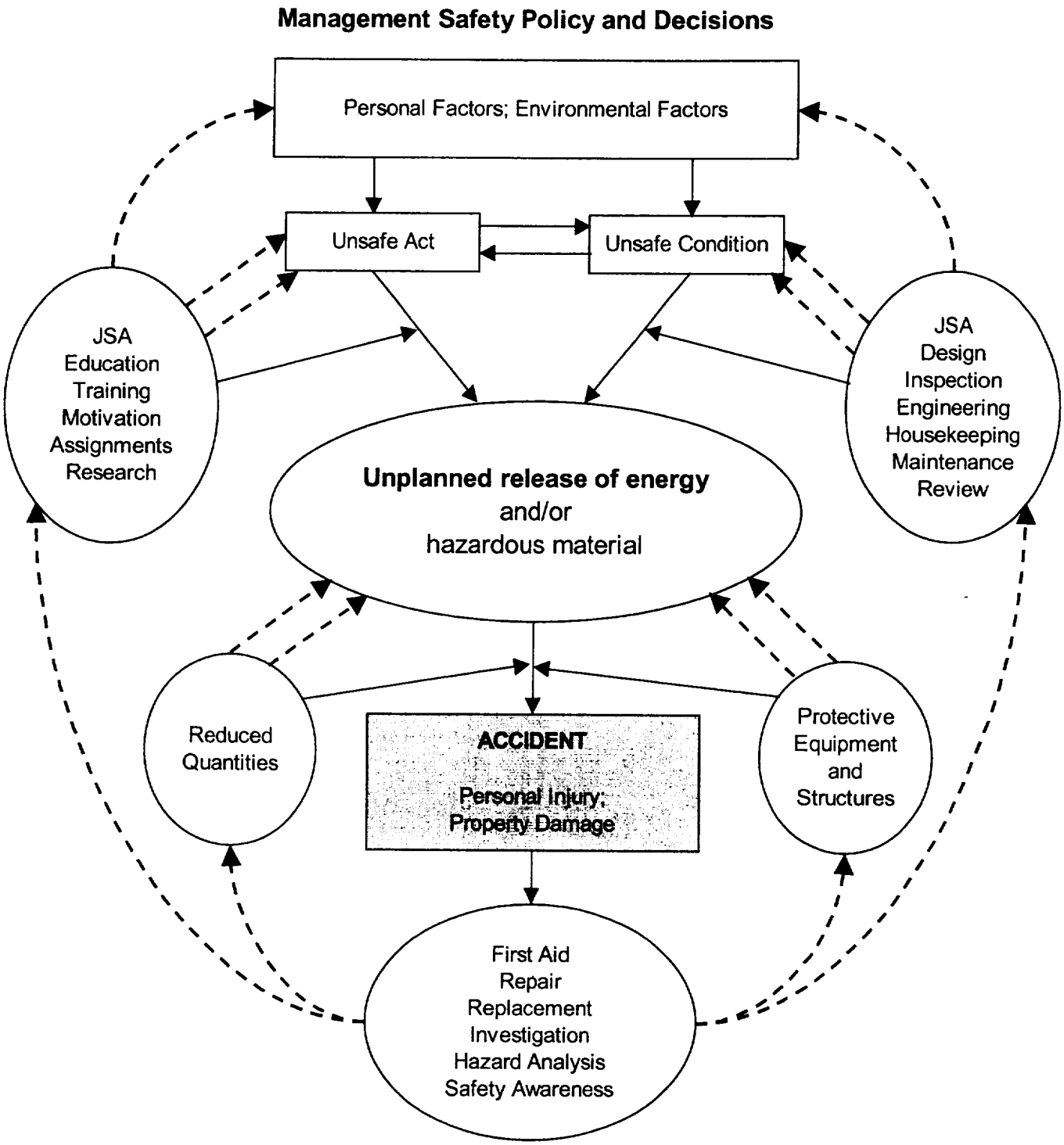
Source: [18]

Figure 4-5: Energy Model (Ball)

Ball identifies two types of accidents: energy transformation and energy deficiency. An energy transformation accident results when a controlled form of energy is converted to another form that negatively affects people or property (for example, an accident in which the chemical energy of fuel transformed into the thermal energy of a destructive fire). An energy deficiency accident happens when the energy required to execute a primary function is not available and the subsequent result is injury to people or damage to property (for example, the destruction of a spacecraft after it loses all electrical power and control functions and then disintegrates in space).

Zabetakis published a fourth update of the Domino theory (see Sections 4.2.1, 4.2.2, 4.2.3) in the Mine Safety and Health Administration (MSHA) safety manual. In this manual regarding accident causation, Zabetakis declared that within the framework of the Domino theory, the direct cause of accidents is an unplanned release of energy (such as mechanical, electrical, chemical, thermal, or ionizing radiation energy) and/or hazardous material (such as carbon monoxide, carbon dioxide, hydrogen sulfide, methane, and water) [18].

Figure 4-6 demonstrates how an unsafe act or an unsafe condition can trigger unplanned releases of energy and/or hazardous material and therefore cause an accident in Zabetakis' energy model.



Source: [18]

Figure 4-6: Zabetakis' Updated Domino Theory / Energy Model

4.4 Systems Theory Models

Systems theory models of accidents adhere to the perspective that accidents arise from interrelationships and interactions among humans, machines, and the environment. Rechtin defines systems as "collections of different things which together produce results unachievable by the elements alone" [20]. Thus, systems theory focuses on the system as a whole and emphasizes that systems, as constructs of related elements, can be studied in the abstract, independent of their context (e.g., software, launch vehicles, communication networks) [21].

According to Hale, the use of systems approach to safety research considers an individual as one of the elements, both human and material, which interact within a defined system boundary to produce a dynamic, adaptive response to that system's environment and to move towards system goals. The human elements of the system differ from the material ones in being (at least potentially) aware of the existence of the system and its goals and being able to plan and carry out their behavior in the light of their predictions about its outcome [9].

4.4.1 Leplat Systems Approach

Leplat defines an accident as a consequence of a dysfunctioning in the system that does not work as planned [17]. He centered his systems approach on the identification and reduction of dysfunctionings of the entire system. He believed that accidents were the result of a network of causes rather than a single cause. Rather than specifying the system variables to be considered or the system components to be investigated, Leplat

emphasized the need to assess problems in systems functioning and production that allowed an accident to occur. He proclaimed that "to think of an accident in terms of a system is to search for the mechanisms which produced it and for the characteristics of the system which may give an account of this process" [17].

Leplat categorized dysfunctionings into two categories [17]:

1. Deficiencies in the articulation of subsystems
2. Lack of link-up between the elements of a system

Pertaining to the deficiencies in the articulation of subsystems, Leplat asserts that the functioning of the system as a whole relies on the functioning of the individual subsystems as well as on the synergy of the functionings towards achieving the system goals. Leplat lists the following articulation factors in the causation of accidents:

- Boundary areas as zones of insecurity (e.g., poorly defined functional responsibilities for departments within an organization).
- Zones of overlapping as zones of insecurity (e.g., conflicts due to two or more departments within an organization affecting the same system element).
- Asynchronous evolution of the subsystems of a system (e.g., a change in one subsystem does not account for its effect on another subsystem).

Poor link-up between elements within a system may be a primary factor leading to accidents. Deficient link-ups among elements or subsystems within a system may

manifest themselves in forms such as poor communication among teams within an organization or non-correspondence of individual capabilities to job responsibilities.

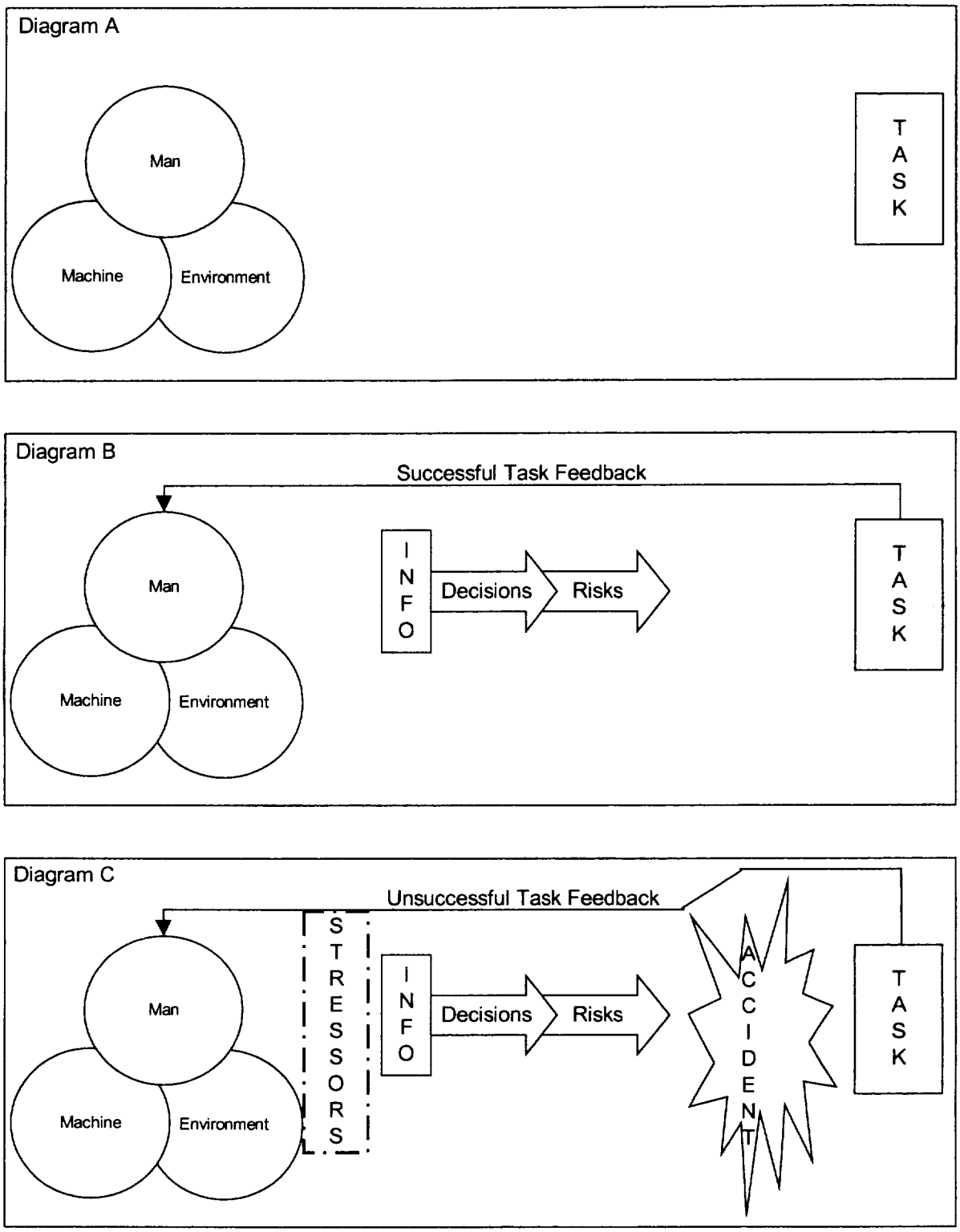
4.4.2 Firenze Systems Model

In 1973, Firenze proposed his systems model that integrates the interactions of the physical equipment, the human who performs tasks with the equipment, and the environment where the process transpires [18].

Diagram A in Figure 4-7 shows the void that exists between the man-machine system and its task. A sequence of processes must occur in order for the man-machine system to reach the system goal.

As depicted in Diagram B of Figure 4-7, man must make decisions based on the various information and data available. With better information, man can make better decisions and reduce the risk involved. With poor or inaccurate information, man is susceptible to flawed decisions and bad risks, which could directly lead to accidents. With the decisions made, man will take risks as he depends on the equipment to perform effectively and the environment to support the functions to attain his objective.

Diagram C in Figure 4-7 illustrates the effect of variables known as "stressors" (of psychological, physiological, or physical origin) that can block man's decision-making capability [18].



Source: [18]

Figure 4-7: Firenze Systems Model

Firenze contends that even with sufficient information, training and capability, man is not perfect and will still make errors under certain circumstances which could lead to an accident. He does state that "chances are that if his decision-making ability is sufficiently developed, along with his comprehension of the hazards connected with his job, and his ability to anticipate and counter accident situations, he stands a better chance of surviving without injuries than if he had no comprehension of the problem at all" [18].

4.4.3 Perrow System Accident Model

Perrow defines a "system accident" or "normal accident" as the unanticipated interaction of multiple failures resulting in damage to subsystems or the system as a whole, disrupting the ongoing or future output of that system [22]. He states that the term "normal accident" is meant to signify that multiple and unintended interactions of failures are unavoidable given the system characteristics.

Perrow proposes that systems can be divided into four levels of increasing aggregation: units, parts, subsystems, and system. Incidents entail the failure of or damage to parts or units only, whereas accidents involve damage at the subsystem or system levels. According to Perrow, component failure accidents happen when one or more linked components (part, unit, or subsystem) in an anticipated sequence fails. Although system accidents start with a component failure, the interaction of multiple failures in unexpected ways is the one of the distinguishing characteristics of system accidents.

4.5 Cognitive Models of Human Error

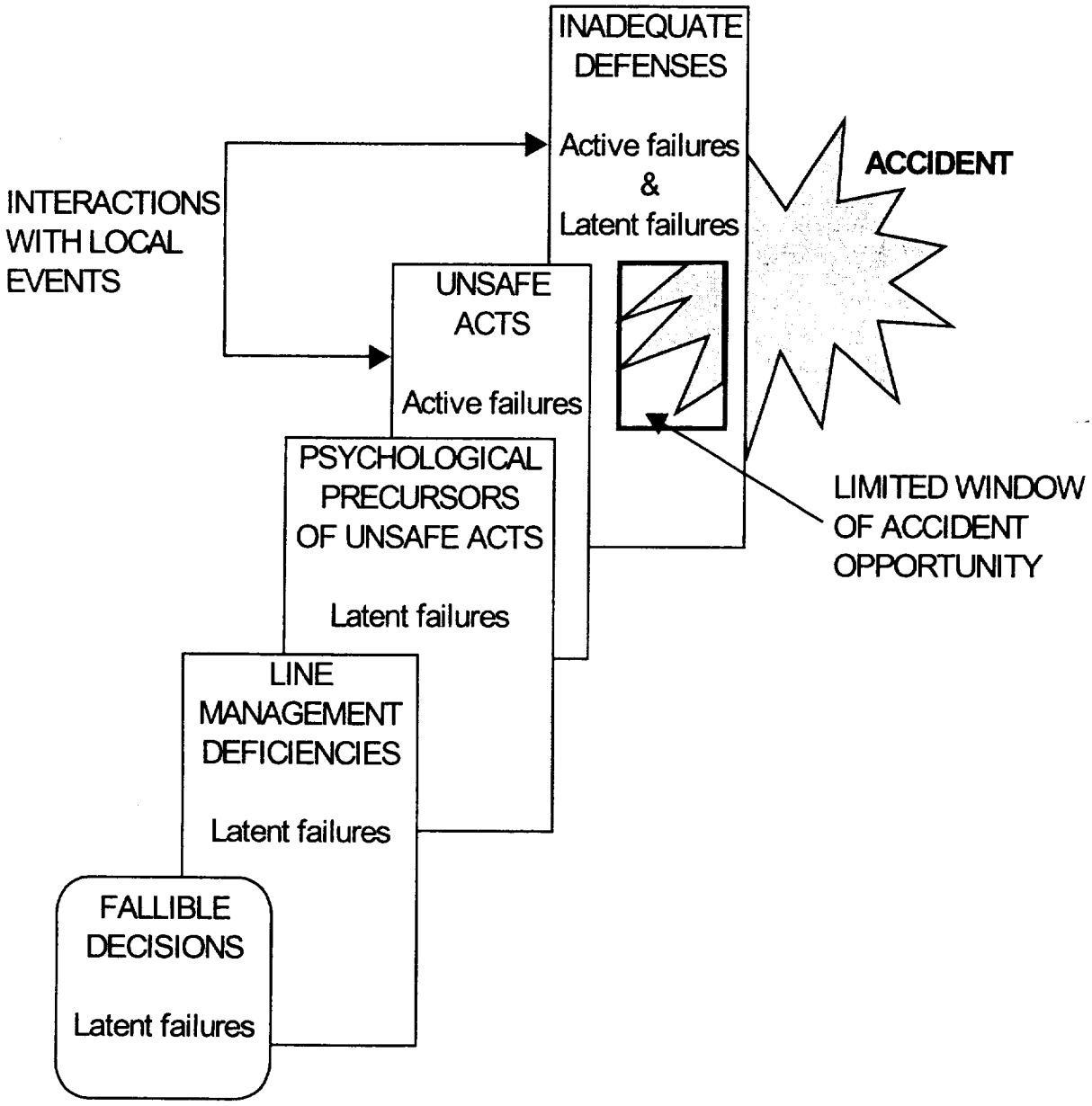
Cognitive models of human error take into account the psychological mechanisms that govern human thought and action. Various models exist that represent the cognition activities of man in performing particular tasks. The models described in this section are cognitive models that focus on the influence of human error in the causation of accidents.

4.5.1 Human Error Contribution to Accident Causation (Reason)

Reason defines "error" as "a generic term to encompass all those occasions in which a planned sequence of mental or physical activities fails to achieve its intended outcome, and when these failures cannot be attributed to the intervention of some chance agency" [16]. In considering human contribution to accidents in complex systems, Reason distinguishes between two types of errors: "active errors" and "latent errors" [16].

Active errors have an immediate impact on the system and are associated with the performance of operators interfacing with the system. Latent errors may lie dormant within the system for a prolonged amount of time and may become evident only when combined with other system factors. Designers, high-level decision makers, and managers are among those responsible for introducing latent errors into a system. Operators are often the inheritors of system defects originating from poor design or bad decisions. Reason claims that latent errors "pose the greatest threat to the safety of high-technology systems" [16].

Figure 4-8 displays how various human errors can contribute to the breakdown of complex systems, resulting in an accident.



Source: [16]

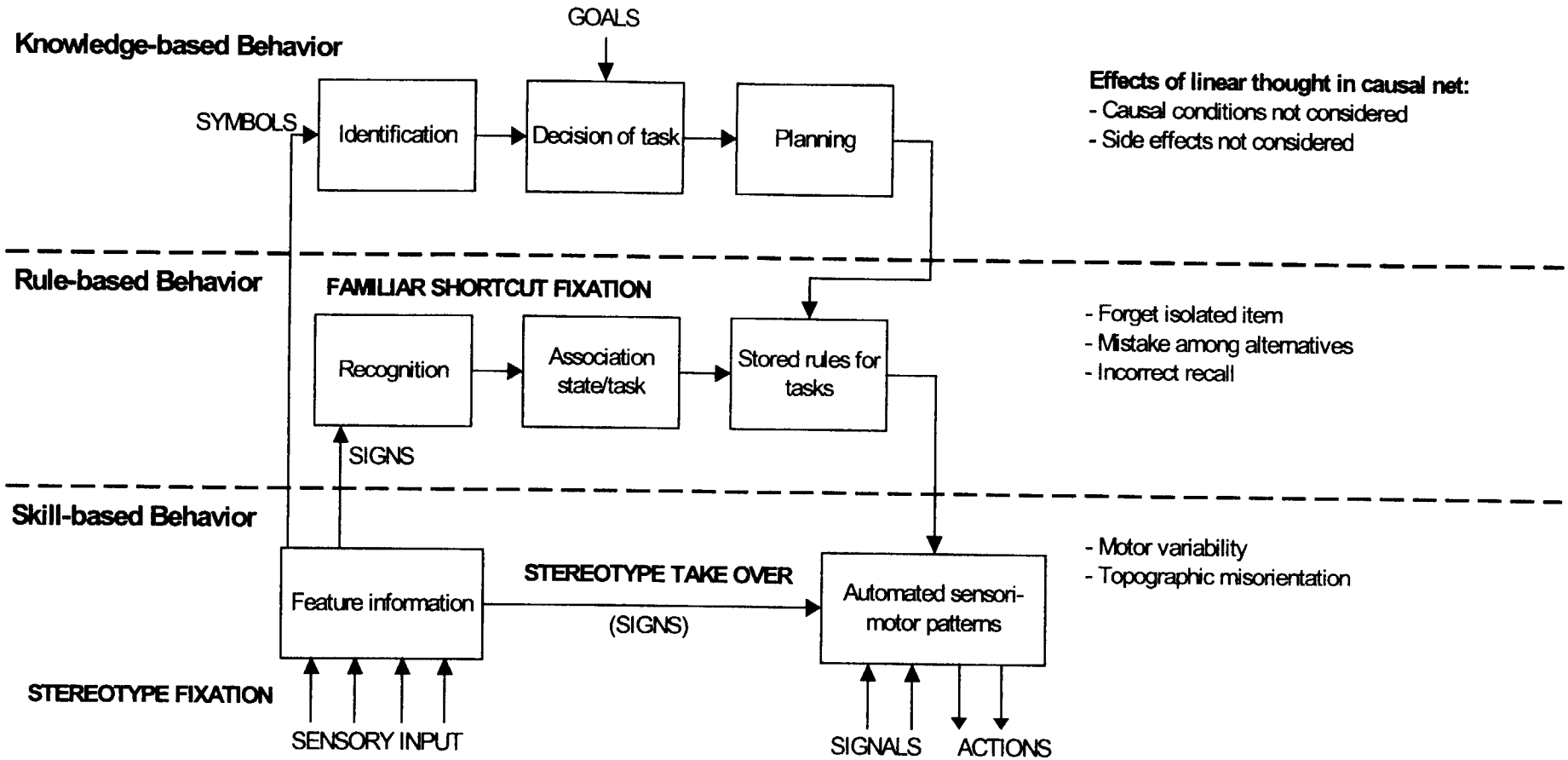
Figure 4-8: Human Contributions to Accident Causation

The premise of Reason's model is that fallible decisions made by system designers and high-level decision makers are the root causes in a system accident. A window of opportunity or weakness opens up for a system accident through the interaction of human contribution, various planes of active/latent failures, and other system factors/events. Thus, human cognition as it pertains to decisions made in the upstream design and management processes plays a major role in the emergence of an accident in a complex system.

4.5.2 Skill-Rule-Knowledge Model (Rasmussen)

Rasmussen's Skill-Rule-Knowledge model represents human cognition and behavior in terms of a hierarchical control structure with three levels: skill-based behavior, rule-based behavior, and knowledge-based behavior [23]. These three cognitive control levels and their relationships are shown in Figure 4-9.

The skill-based behavior level denotes the sensory-motor performance of tasks achieved without conscious control or attention. The sensed information at this level is perceived as signals, which are continuous and quantitative indicators of the time-space behavior of the environment [17]. These signals are processed purely as physical time-space data. Human errors can appear at this level when the sensory-motor control is inadequate, the internal model becomes unsynchronized with the environment, or the schema control changes unintentionally.



Source: [17]

Figure 4-9: Skill-Rule-Knowledge Model (Rasmussen)

At the rule-based behavior level, the information is perceived as a sign if it proceeds to trigger or change predetermined actions. These signs refer to behavior from previous occurrences or by convention. Hence, the control at this level is teleologic in that the rule or control is selected from earlier successful experiences [23]. Errors can arise when the human recalls or activates rules incorrectly or when the human can not adapt properly to system changes.

Whereas procedural knowledge is the focus for rule-based behavior, structural knowledge and mental representations of system operations form the foundation for knowledge-based behavior. The information is perceived as symbols, which refer to internal, conceptual representations. Cassirer notes that "signs and symbols belong to two different universes of discourse: a sign is part of the physical world of being, a symbol is part of the human world of meaning" [23]. This knowledge-based behavior entails establishing a goal and planning interactions with the environment.

At this functional reasoning level, the opportunity for error can occur in situations where adaptation can not be accomplished due to limited knowledge/information and where bad decisions negate achievable adaptation. As stated by Mach back in 1905: "Knowledge and error flow from the same mental sources, only success can tell the one from the other" [23].

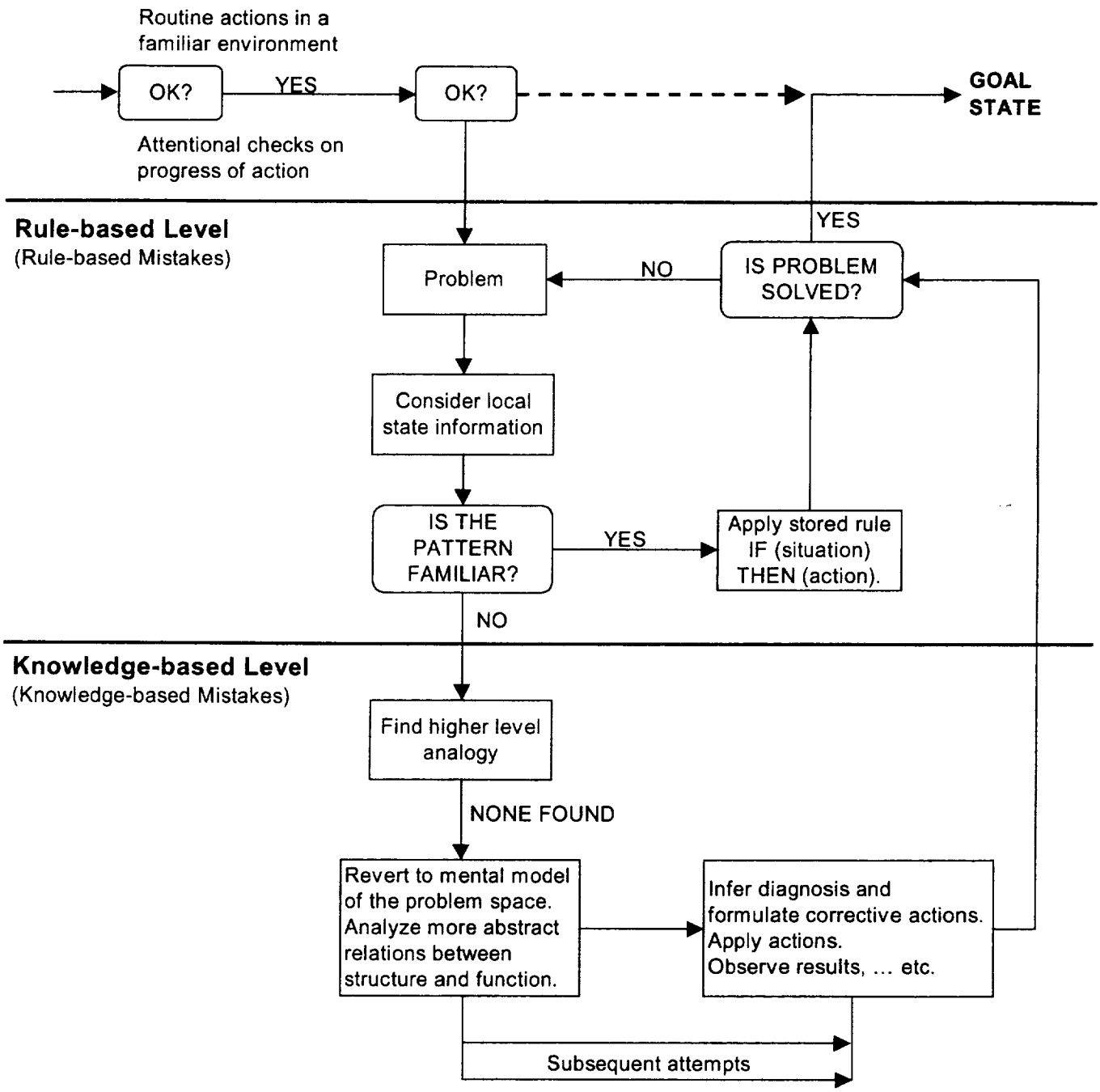
4.5.3 Generic Error-Modelling System (Reason)

Reason developed the Generic Error-Modelling System (GEMS) as a cognitive framework for locating various forms of human error. GEMS attempts to integrate two types of errors [16]:

1. Slips and lapses, in which actions deviate from intention as a result of execution failures or storage failures.
2. Mistakes, in which actions may run according to plan but the plan is inadequate in terms of attaining its preferred outcome.

Although based on Rasmussen's Skill-Rule-Knowledge model, GEMS provides a more integrative model of error mechanisms operating at all three levels of skill-based, rule-based, and knowledge-based performance (see Figure 4-10).

The operations for GEMS fall into two categories: those preceding problem detection (skill-based) and those following the problem (rule-based and knowledge-based). Reason asserts that the key feature of GEMS is that, when confronting a problem, human beings are strongly biased to establishing (via automatic pattern matching at the rule-based level) whether or not local indications have been previously encountered as opposed to resorting to the more effortful knowledge-based level (even where the latter is demanded from the outset) [16]. Errors at the skill-based level can be attributed primarily to monitoring failures whereas mistakes at the rule-based and knowledge-based levels are coupled with problem solving.



Source: [17]

Figure 4-10: Generic Error-Modelling Systems (GEMS)

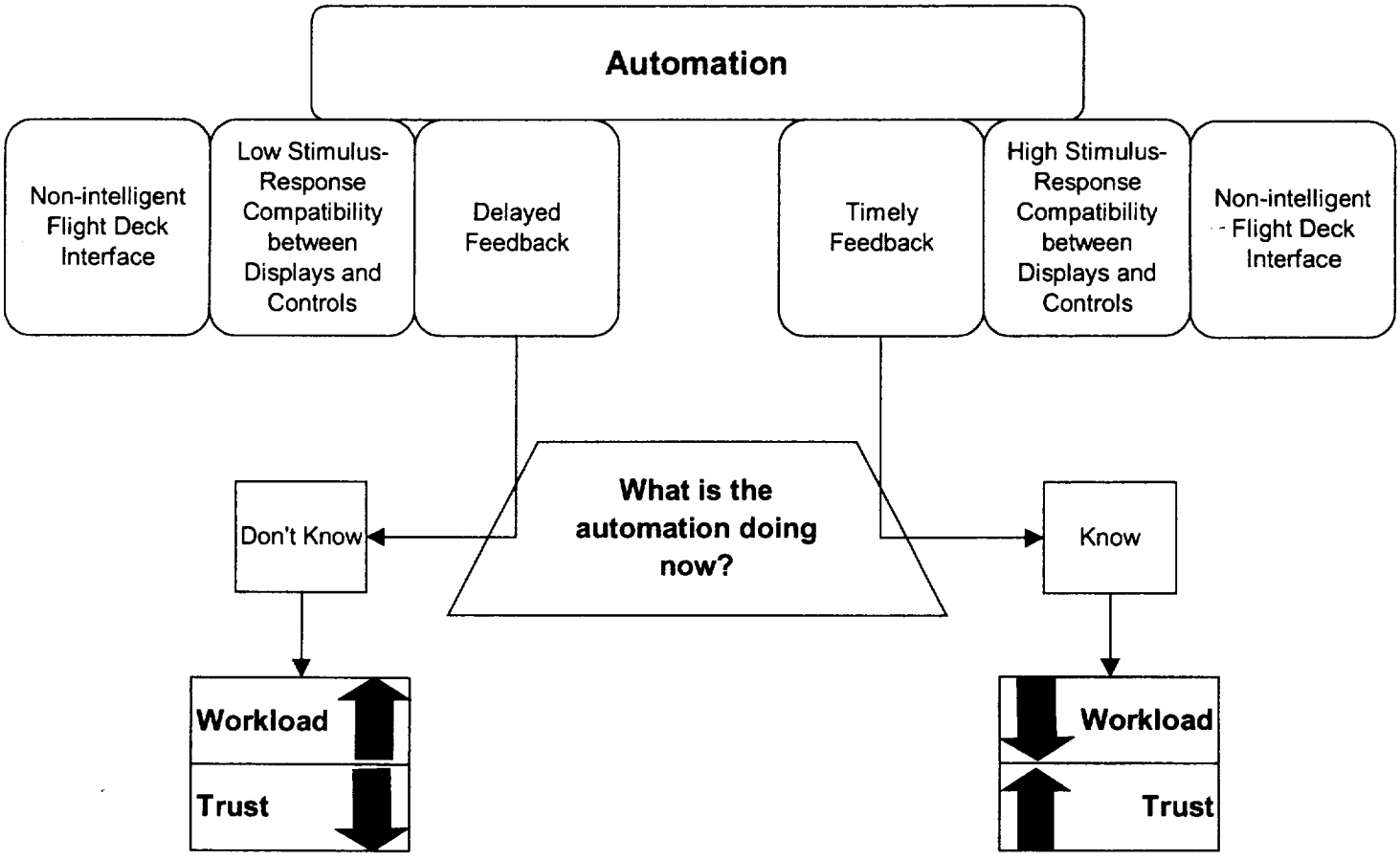
4.2.6 Internal Model of Operator (Kantowitz and Campbell)

Kantowitz and Campbell have researched the relationship between the operator and automated systems with respect to the efficiency and safety of man-machine systems. They have defined an internal model of the operator to be the operator's internal representation and understanding of elements, dynamics, processes, inputs, and outputs of a system [24]. The operator utilizes this internal model as an organizing schema for planning various activities, hypothesizing about system component relationships, and executing system tasks. The operator's success in accomplishing the goal of a specific system task depends upon the accuracy and functionality of the internal model. Kantowitz and Campbell also stress the importance for a "match" to exist among the operator's internal model of the system, the operational characteristics of the system, and the designer's model of the system.

Applying their theory to automation in aviation systems, Kantowitz and Campbell suggest that designers for automated flightdecks can integrate pilots' mental models into their system design in the following ways [24]:

- Match task demands to environmental demands.
- Provide timely and accurate feedback to the pilot.
- Design control configurations and display formats that are consistent with both system performance and pilots' expectations for the system.
- Develop training strategies that facilitate the development of appropriate internal models.

Interfaces to man-machine systems as well as the relationship between a system stimulus (e.g., a display) and a system response (e.g., a control action) greatly influence the operator's cognitive representation of the system and its components. As depicted in Figure 4-11, the combination of the varying aspects of system interface design, stimulus-response (S-R) compatibility, and feedback alters the operator's workload, trust, and internal representations of the automated system.



Source: [24]

Figure 4-11: Internal Model of Operator in Automated Systems

The traditional accident models discussed in this chapter show the diverse methods to represent and explain accidents. Each of these accident models has varying strengths and weaknesses. Applied individually, each of them cannot comprehensively explain the complex system accidents of today. The next chapter proposes a completely different and new methodology for obtaining the appropriate data for effective risk assessment and detecting potential problems before accidents can occur.

Chapter 5

Model-based Framework for Risk Assessment

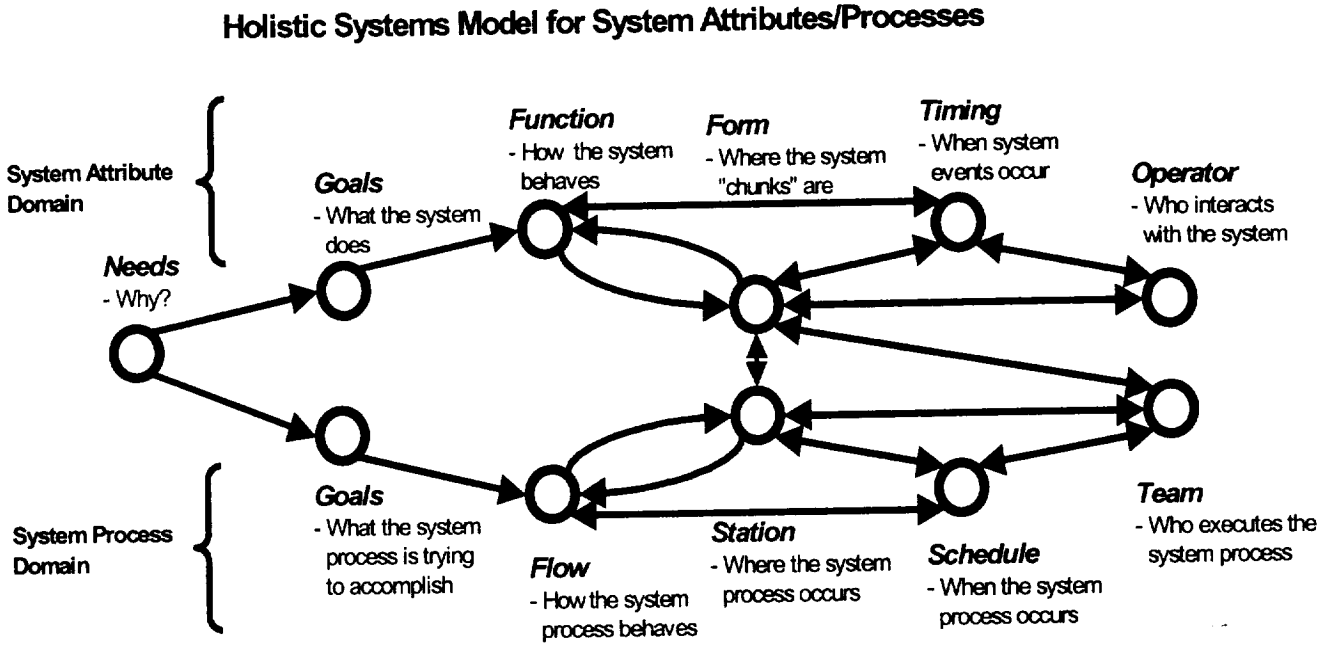
5.1 Overview

The framework proposed in this chapter is centered around the use of top-down hazard analysis techniques in the upstream phases of development in order to identify hazards and then eliminate or control them in the design of complex systems. The underlying accident model for this framework is a holistic, systems theory-based model. Using top-down hazard techniques, this framework facilitates the association of system errors to human-induced errors, component interaction problems, software defects, or organizational inadequacies within the context of the holistic systems model.

5.2 Holistic Systems Model

The underlying systems model utilized in this framework is adapted from Crawley's model of the Total Holistic View of Product/Process Architecture [25]. As shown in Figure 5-1, the model represents system attributes in the system attribute domain as follows:

- **Need:** Purpose (Why the system is built)
- **Goal:** Performance (What the system does)
- **Function:** Behavior (How the system behaves)
- **Form:** Structure (Where the system "chunks" are)
- **Timing:** Action (When system events occur)
- **Operator:** Users (Who interacts with the system)



Source: Adapted from Crawley [25]

Figure 5-1: Holistic Systems Model for System Attributes/Processes

In the system process domain, the model represents system development processes as follows:

- **Need:** Purpose (Why the system development process is in place)
- **Goals:** Performance, timeframe, cost, risk (What the system development process is attempting to accomplish)
- **Flow:** Workflow analysis (How the system development process behaves)
- **Station:** Design station (Where the system development process occurs)
- **Schedule:** Plan (When the system development process occurs)
- **Team:** Organization (Who executes the system development process)

This systems model incorporates a holistic approach by considering all influences and consequences of factors interacting with system attributes and processes. This approach nourishes viewpoints of totality and promotes completeness for system design and operations. With respect to system accidents or failures, this approach evaluates the interactions among the system attributes and processes as a whole as opposed to evaluating failures within individual system components. Accidents in complex systems today are occurring where individual system components are functioning according to specifications, yet the overall system is malfunctioning.

5.3 Upstream Influence Considerations

Plato stated back in 4th Century B.C.: "The beginning is the most important part of the work" [20]. As depicted in Figure 5-2, the holistic systems model accounts for technological, organizational, managerial, regulatory, and safety influences, among others, on the early phases of system development.

Market conditions and competition are key factors that influence the user/customer wants and needs for a system. These factors shape the customer requirements for a system. From market research and competitive analysis, a system architect can transform customer desires and expectations into a set of directives to guide the system design processes further downstream.

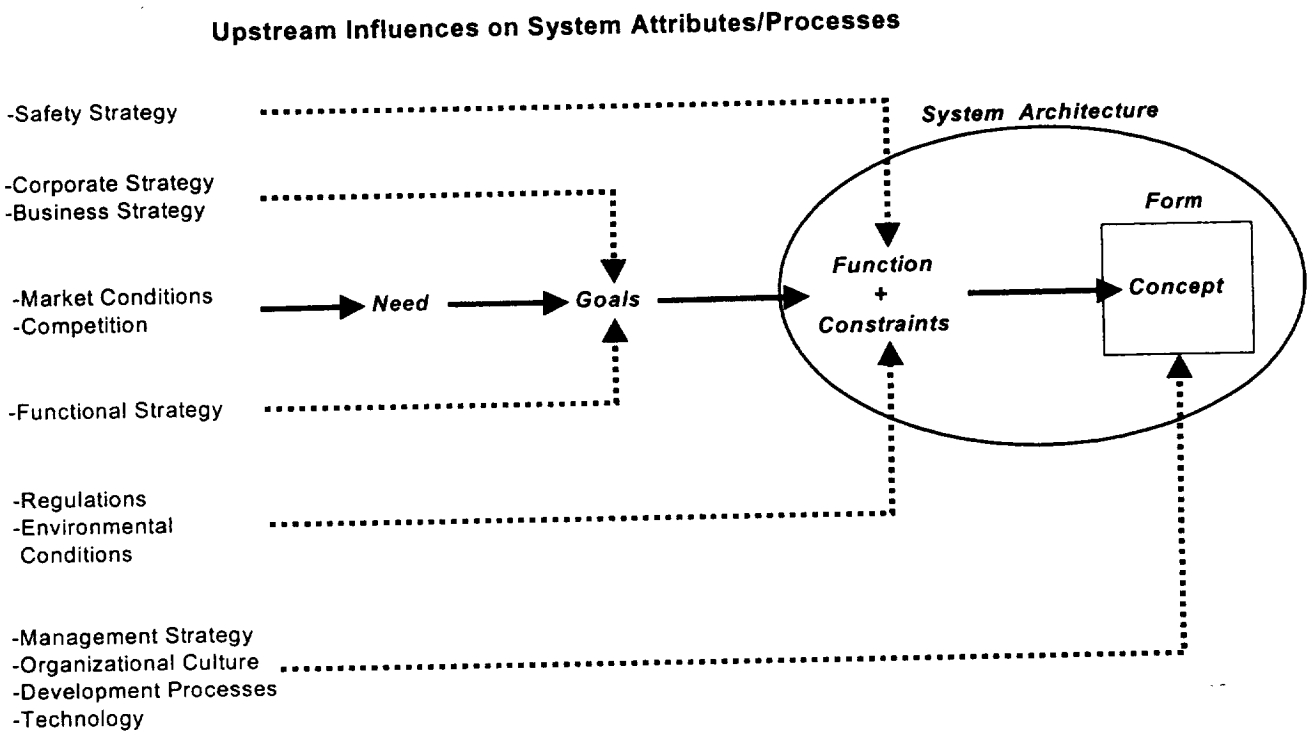
The customer needs are interpreted to define system goals. Corporate strategy, business strategy, functional strategy, and other business environment aspects play significant roles

in the formulation of system goals as they are documented in system specifications. An important step is to adequately document the design rationale or intent of the system specifications.

System functions are derived from the established system goals. Safety strategy and regulations are upstream factors that greatly affect the system functions. Compliance with safety policies, federal/international regulations, socioeconomic policies, and standards must be considered.

System form is the structure of the physical/logical embodiment of the system functions. Organizational culture, management strategy, and operations strategy are some of the upstream influences that impact the development of the system form. The selection of technology strategy with respect to the deployment of new technology or the reuse of existing technology is a principal decision.

System concept is the system vision or idea that maps system function to system form. System concept can also be characterized as the specification of the list of design parameters that, when specified, will define the design. System function and system form are iterated through conceptual design to allow the execution of all system functions.



Source: Adapted from Crawley [25]

Figure 5-2: Upstream Influences on System Attributes/Processes

5.4 New Holistic Systems Accident Model

A new holistic systems accident model can be derived from the previously described holistic systems model with consideration to the upstream influences on system attributes and processes.

As presented in Figure 5-3, the holistic systems accident model illustrates the potential conditions, factors, and influences on system attributes and processes that can cause an accident or incident.

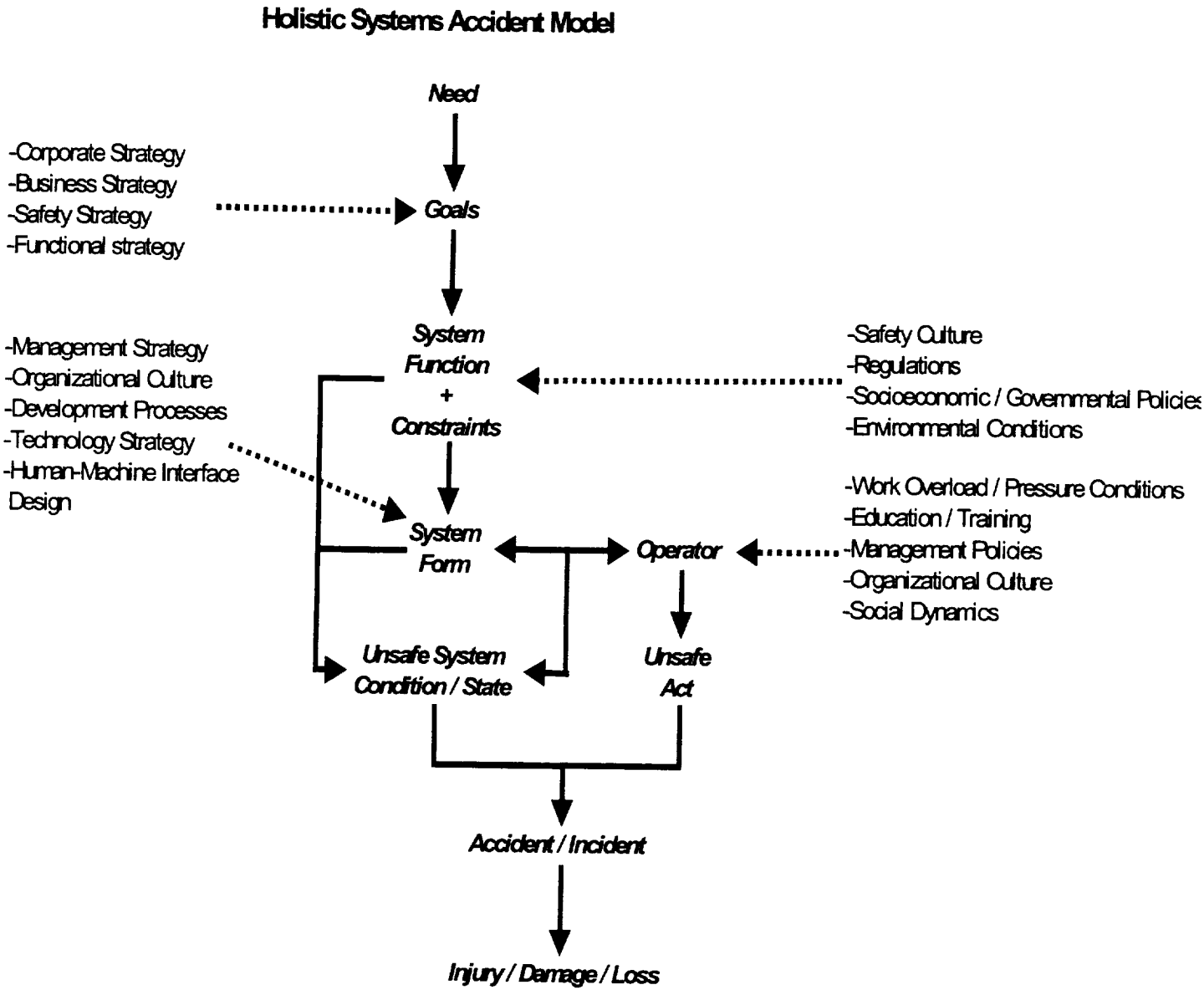


Figure 5-3: New Holistic Systems Accident Model

The influences of corporate strategy, business strategy, safety strategy, and functional strategy shape the definition of system goals. Incomplete or inadequate upstream strategies can lead to system accidents as they directly affect the functionality, quality,

and performance of the system as well as set the expectations of system operators and end users.

Governmental regulations, socioeconomic policies, and environmental conditions are among the potential influences that can increase the risk of an accident. Budgetary restrictions and cost-cutting measures can place undesirable constraints on the developed system. A negligent safety culture can lead to system functionality with hazardous effects.

The system can transition into an unsafe condition or state due to poor management strategy/policy, adverse organizational culture, flawed system development processes, inappropriate technology strategy, or excessive complexity in human-computer interface design. The effectiveness of the management and execution of system processes along with the type of culture instilled within the organization developing the system are strong indicators of whether or not an accident is likely to occur for that system.

Unsafe acts or bad decisions by the operator also increase the risk of a system accident. Work overload, high-pressure work environments, and insufficient education/training are some of the influential factors in causing an operator to perform an unsafe act or make a decision to err. Poor management procedures and detrimental organizational culture can give rise to unsafe system operations, which can cause a system accident.

This new holistic systems accident model is considerably different than the accident models described in Chapter 4. By incorporating influences and factors on multiple

hierarchical levels, this new holistic systems accident model takes a comprehensive approach in modeling accident causality from managerial, organizational, regulatory, educational, environmental, economic, technical, and human cognition perspectives.

Most of the accident models concentrate on a specific aspect of accident causality. The process accident models (domino theory, multilinear events sequence, chain-of-events) focus on the sequence of events and factors contributing to an accident, but they do not account for all of the hierarchical levels of causality. The energy models (Ball, Zabetakis) are centered on the unplanned release of energy and/or hazardous material, but they do not investigate further into the systemic root causes of the energy release. The cognitive models of human error (Reason, Rasmussen, Kantowitz and Campbell) take the core approach of examining psychological mechanisms that control human cognition and behavior and that cause human error or human contribution to accidents.

Although the Leplat systems approach, Firenze systems model, and Perrow systems model are based on systems theory, they do not stress the importance of upstream influences on the various stages in the design and development life-cycle for the system. Additionally, the new holistic systems accident model probes into specific system attributes and system development processes and their effect on accident causality.

5.5 New Risk Assessment Framework

Utilizing the new holistic systems accident model as a foundation, the following five system safety steps comprise the key components for a risk assessment framework in which to provide a measure of safety in the design and operation of human-computer controlled systems:

1. Preliminary Hazard Analysis on System Functions
2. Fault Tree Analysis of System Function Hazards
3. Safety Design Constraint Identification for System Function Hazards
4. Mitigation Feature Assessment for System Function Hazards
5. Human-Machine Interface Design for System Form

The new holistic systems accident model affects how each system safety step is performed by evaluating the new systemic view of accident causality through the multiple hierarchical perspectives.

5.5.1 Preliminary Hazard Analysis on System Functions

The first step in this framework involves a Preliminary Hazard Analysis (PHA) conducted on system functions early in the development process to identify potential hazards of the system and exposures at the system boundaries. Identification of hazards during upstream processes is crucial in order for the system architect to effectively derive the system concept and form.

Leveson recommends the following steps for PHA [2]:

1. Determine potential hazards that might exist during operation of the system and their relative magnitude.
2. Develop guidelines, specifications, and criteria to follow in system design.
3. Initiate actions for the control of particular hazards.
4. Identify management and technical responsibilities for action and risk acceptance and assure that effective control is exercised over the hazards.
5. Determine the magnitude and complexity of the safety problems in the program (i.e., how much management and engineering attention is required to minimize and control hazards).

The new holistic systems accident model steers the PHA to utilize a systemic approach in its consideration of upstream influences on system attributes and system processes for ascertaining potential system function hazards.

5.5.2 Fault Tree Analysis of System Function Hazards

For each significant hazard identified in the PHA, the framework utilizes the top-down approach of a fault tree analysis (FTA) to analyze the causes of the system function hazards. With initial assumptions made about the system state and environmental conditions, the system architect determines lower level causal events and relationships associated with the top level hazard event through the construction of fault trees (as was shown in Figure 3-1 in Chapter 3). The analysis continues down each level of the fault tree until a primary event is attained. The new holistic systems accident model guides the analysis by assessing each level of the fault tree with consideration to human, technical, management, organizational, and cultural influences, among others.

5.5.3 Safety Design Constraint Identification for System Hazard Functions

Once the pertinent fault trees are constructed, the framework then entails the identification of safety design requirements and constraints. The designer/analyst must design safety into the system through the elimination, reduction, or control of the identified system function hazards. Developing safety criteria for minimal damage is also another important consideration for safety constraints.

Leveson provides the following useful categorization of safe design techniques (in order of their precedence) [2]:

1. Hazard Elimination
    - Substitution
    - Simplification
    - Decoupling
    - Elimination of specific human errors
    - Reduction of hazardous materials or conditions
2. Hazard Reduction
    - Design for controllability
    - Barriers (lockouts, lockins, interlocks)
    - Failure minimization (redundancy, safety factors and safety margins)
3. Hazard Control
    - Reducing exposure
    - Isolation and containment

- Protection systems and fail-safe design

4. Damage Reduction

After the initial safety design constraints are established, the system architect must make refinements to the constraints through an iterative process. The impact of governmental regulations, socioeconomic policies, and environment conditions must be considered with respect to the new holistic systems accident model during the identification process of the safety design constraints.

5.5.4 Mitigation Feature Assessment for System Hazard Functions

With the safety design constraints identified, the framework progresses to an assessment of mitigation features to reduce the likelihood of the system function hazard leading to an accident. Simple designs, well-defined interfaces, and intuitive procedures for operators are conducive to improving system safety. Physical interlocks and software controls are additional features that can be incorporated into the system design to facilitate fail-safe operations. Using the new holistic systems accident model, mitigation features must be assessed and implemented such that unsafe system conditions and hazardous system functions are not manifested through the system.

5.5.5 Human-Machine Interface Design for System Form

Rechtin states that "the greatest leverage in architecting is at the interfaces" [20]. The design of the human-machine interface (HMI) shapes the form of the system. By applying

design principles with respect to attaining efficient human-machine interactions and reducing safety-critical human errors, safety-enhancing system functions can be made straightforward to achieve and unsafe system functions difficult to achieve. Upstream considerations pertaining to technology strategy, management policies, organizational culture, and system development processes must be weighed via the new holistic systems accident model to successfully develop the system interfaces.

Leveson recommends the following process for designing a safer HMI: [2]

- 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.

In addition to the Preliminary Hazard Analysis performed at the system level in the beginning of this framework, another hazard analysis should be performed on the HMI itself. Analysis of potential hazards for the HMI must be done during the design phase of the HMI, not after its completion. Results from this hazard analysis along with feedback from other resources (e.g., accident reports) should be folded back into the HMI design and refined through this iterative process.

As the next chapter will show, this holistic model-based framework for risk assessment can be applied to real world, human-computer controlled system applications to achieve a risk reducing, safety-conscious system design.

Chapter 6

Case Study for Model-based Framework: MAPS

6.1 Overview

The case study presented in this chapter shows how the previously described model-based framework can be applied to a particular system called the mobility and positioning software (MAPS) system. As documented in Appendix C, MAPS controls a robot intended to service the heat-resistant tiles on the Space Shuttle by positioning it under the appropriate place under the spacecraft and by moving it around the hangar. The team of Demerly, Hatanaka, and Rodriguez conducted the MAPS analysis and design effort for this case study [27].

6.2 MAPS Needs and Goals

Based on the MAPS information provided in Appendix C, the customer need is for a mobility and positioning software system for a tessellator robot that can service tiles on the Space Shuttle.

Employing the holistic systems accident model, the case study assesses the impact of corporate strategy, business strategy, safety strategy, and functional strategy on the development and operations of the MAPS system. Safety is the primary focus for this case study. Thus, the system goal for the system architect in this case study is to design a safe human-computer controlled system within which the tessellator robot can be operated to accomplish the required tasks.

6.3 Preliminary Hazard Analysis on System Functions

By analyzing the various MAPS system functions in Appendix C and evaluating the system boundaries, the following system hazards were identified:

- Mobile base runs into object
- Robot does not deploy stabilizer legs when moving manipulator arm
- Mobile base moves with stabilizers down
- Manipulator arm hits something

The consideration of environmental conditions from the holistic systems accident model directly resulted in the identification of PHA system hazards pertaining to the robot hitting obstacles within the environment.

6.4 Fault Tree Analysis of System Function Hazards

A fault tree was constructed for each of the four top-level system function hazards identified in the PHA. The top level system function hazard was decomposed into levels representing intermediate causal events and repeated until bottom levels were reached that signified the primary causes. At each level of the MAPS FTA, the holistic systems accident model was applied to assess the impact of multilevel, upstream factors that could potentially spawn hazardous system conditions or unsafe operator interactions. Technology strategy (sensors), environmental conditions (gravitational factors, obstacles), and human-machine interface design (feedback mechanisms) were some of the aspects analyzed.

Figure D-1 Part 1 and Part 2 in Appendix D show the first fault tree for the top level system function hazard of when the mobile base runs into an object. MAPS logic errors and mobile base operational defects were some of the primary causes (identified by circles at leaf nodes in the fault tree) that were discovered.

The system function hazard of the robot not deploying the stabilizer legs when moving the manipulator arm was investigated in the second fault tree in Figure D-2 in Appendix D. Primary causes were attributed to planner errors, sensor problems, and MAPS logic errors. Note that non-MAPS related hazard events were not further decomposed in this case study.

The third fault tree in Figure D-3 in Appendix D illustrated the system function hazard of the mobile base moving without the stabilizer legs down. Software and sensor problems were among the root causes.

The system function hazard of the manipulator arm hitting something was analyzed in the fourth fault tree in Figure D-4 in Appendix D. MAPS control problems and manipulator arm errors were some of the basic causes for this hazard.

The following assumptions were made for the MAPS FTA:

- Non-MAPS related events are not further decomposed
- Planner controls Manipulator Arm
- Planner checks state of legs before moving arm

- MAPS processes intended/actual position data
- Operator overrides Planner commands
- Operator/Planner command conflict resolved in MAPS

6.5 Safety Design Constraint Identification for System Function Hazards

After analyzing data available from the PHA and FTA, the following initial safety design constraints and requirements were identified for the system function hazards:

System Function Hazard	Design Constraint / Requirement
Mobile base runs into object	**Mobile base shall not run into objects** • Mobile base shall move only when commanded • Mobile base shall stop when commanded • Mobile base shall move to correct position • MAPS shall help operator by providing status information back to operator • Mobile base shall avoid objects that enter its path
Robot does not deploy stabilizer legs when moving manipulator arm	**Manipulator arm shall move only after stabilizer legs have been deployed** • MAPS shall deploy legs before manipulator arm moves • MAPS shall not retract legs while manipulator arm is moving • Stabilizer legs must receive commands from MAPS
Mobile base moves with stabilizer legs down	**Mobile base shall move only when stabilizer legs are up** • Mobile base shall not move before stabilizer legs are raised • MAPS shall not deploy stabilizer legs while mobile base is moving • MAPS should be able to respond safely to a conflict of order

Manipulator arm hits something	**Manipulator arm shall not hit other objects** • Manipulator arm shall move only when commanded • Manipulator arm shall move only when base is stopped • Mobile base shall not move without manipulator arm stowed • System shall detect object moving into path of manipulator arm

Through multiple iterations of refinement on the design constraints and requirements, the final set of safety design constraints was established as shown in Appendix E. Using the holistic systems accident model, design constraints were developed and validated against industry regulations and standards. Assumptions were made with respect to core competencies of the development team for the system and budgetary limitations for the technology selection of robot components.

6.6 Mitigation Feature Assessment for System Function Hazards

For each of the safety design constraints and requirements identified, mitigation features were assessed and resolved at the system level, inter-subsystem level, and individual subsystem level. Applying the holistic systems accident model, the case study examined essential upstream factors that included fail-safe strategies, operator culture, and interface design in the creation of system mitigation features at the various system and subsystem levels.

6.6.1 System Level Mitigation

Fail-safe processing modules, design for controllability, elimination of specific human errors, and monitoring were some of the system level mitigation features assessed.

In consideration of fail-safe processing modules, the following questions were posed:

- How can one create correct mechanisms that will allow the system to fail-safe in case the sensors fail?
- How can one design the software so that it will be robust enough to assimilate all of the limitations?

An emergency stop switch (i.e., "deadman switch") would be a key design requirement for a fail-safe system. System level checks and sensor checks would allow for a safe system design. The following sensor checks were considered essential:

- Check if the sensors are ready for operation.
- Check to see if the appropriate input is being provided.
- Check whether or not the output being generated is desired.
- Check for a threshold limit being reached.
- Check for an abnormal value.

With respect to design for controllability, the use of incremental control and intermediate stages were key aspects that were analyzed. By utilizing an incremental control system for the mobile base, the operator would be able to correct wrong positions, receive feedback from the system, test the validity of his/her own mental model, and take corrective actions before serious damage is done. At intermediate stages, MAPS should provide the operator

with options in case of failure. Designers should determine the minimal set of functions required for reaching the system safety state.

Human factors issues for the system were examined in the attempt to eliminate some specific human errors. Compatibility between the display and robot motion was investigated in terms of inside-out views and outside-in views. Stress, workload, and training issues as well as feedback types were considered.

Monitoring mechanisms and problem detection were also decided upon at the system level. Audio and visual alarms (e.g., bells, lights) were found to be important system feedback mechanisms in the system design. In order to detect problems as soon as possible, a good control system must be established. Some mitigation features included implementing assertions in positioning, velocity, and acceleration as well as deploying interlocks to assure success in sequence of events.

6.6.2 Inter-Subsystem Level Mitigation

For inter-subsystem level mitigation, both hardware and software interlocks were proposed. Contact switches on the stabilizer legs as well as on the manipulator arms were the key hardware interlocks between the subsystems. Software interlocks were required for situations during mobile base movement and arm/leg deployment.

6.6.3 Individual Subsystem Level Mitigation

The hardware subsystem and software subsystem needed mitigation features as well. Hardware saturation on MAPS outputs, size/torque of hardware motors, passive braking system, and subsystem "fail into" state were the hardware mitigation features. Software assertions for expected values, software maximum limits, internal state models, and verification/validation via redundant signals were some of the software mitigation features.

For design constraints, mitigation features, and design tradeoffs, the system architect must perform analysis at the system, inter-subsystem, and individual subsystem levels. Appendix E contains a comprehensive design constraint and mitigation feature analysis for MAPS. This MAPS design constraint analysis is segmented by the primary hazard event with its corresponding safety design constraints and proposed mitigation features. Some pertinent notes are also listed where applicable. The final section of the MAPS design constraint analysis documents the system level checks and fail-safe scenarios.

6.7 Human-Machine Interface Design for System Form

The design of the Human-Machine Interface (HMI) comprised of three primary components: visual displays, operational controls, and system state indicators. These components were structured in a control panel configuration as shown in Figure 6-1.

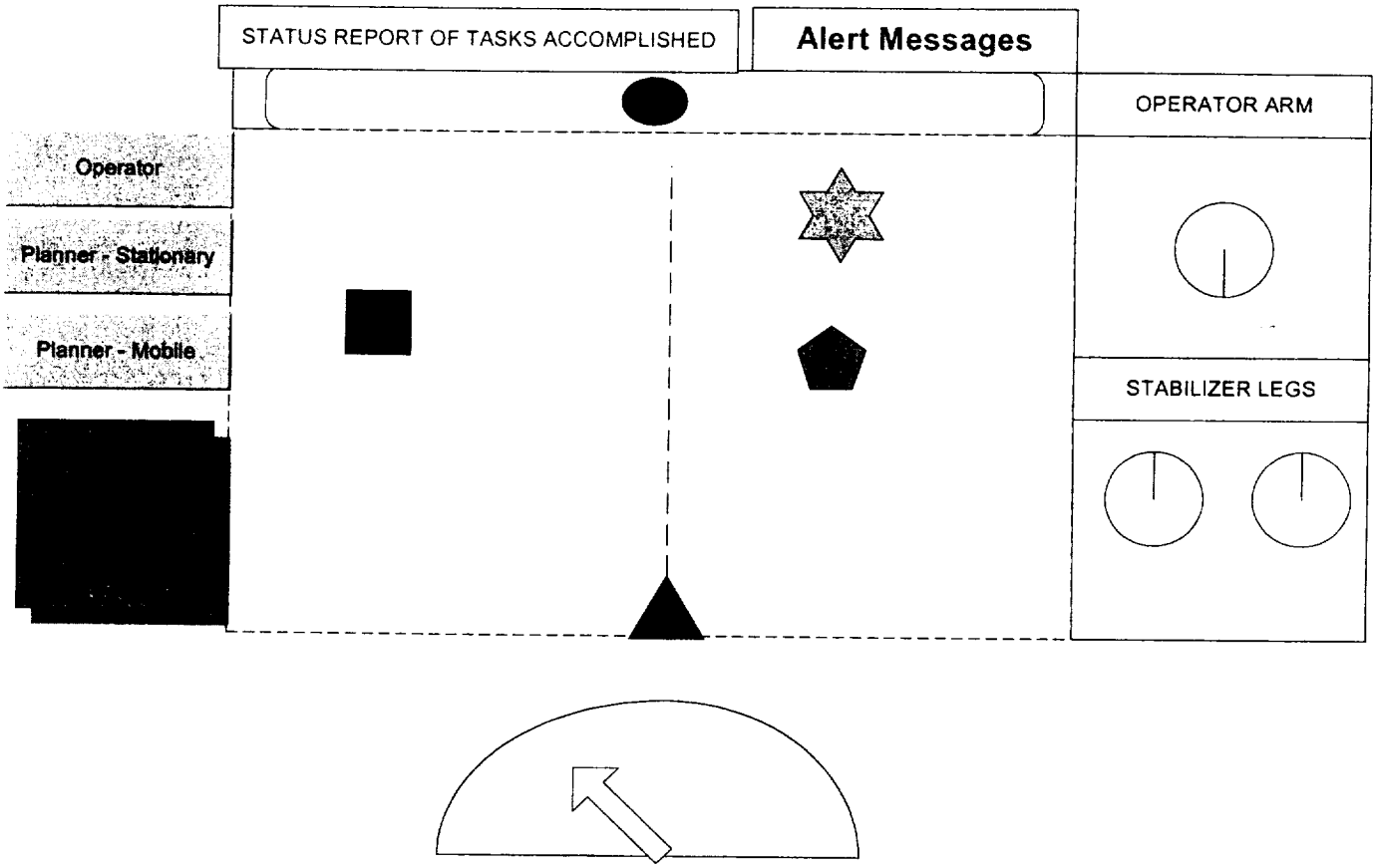


Figure 6-1: MAPS Control Panel Displaying the Plan View

The case study incorporated the holistic systems accident model at this phase to properly consider operator cognition, operator workload factors, feedback controls, interface usability, and training requirements. Operator culture and emergency conditions were also analyzed. The recognition of the causal factors from the holistic systems accident model provides the motivation to set human-machine interface design goals to eliminate unsafe system conditions and unsafe operator actions.

6.7.1 Visual Displays

Visual displays were designed to assist the operator in controlling the robot with respect to positioning and velocity. The following two views were considered essential:

- Camera view
- Plan view

The camera view would be the image from a camera mounted on the robot. This view would show the operator what is in front of the robot, giving the operator the perspective of being on the robot while driving. This view would eliminate the issue of command confusion when the robot is driving toward or when the robot is driving away from the operator because the operator would have the perspective of the robot. Obstructions between the operator and the robot would also be detectable through the camera view.

The plan view would provide a top/plan view of the robot with respect to its environment. In this view, the objects detected by the collision avoidance system would be displayed. In addition, the present trajectory of the robot would be displayed on the screen to show

where the robot would be moving in relation to the other objects. Based upon this information, the relative risk of collision with the objects could be assessed and the appropriate alarms could be enabled.

The camera view and plan view would display much of the same information but in different formats and from different sources such that they could be used to confirm each other. Both views would provide information in easily understandable formats that follow norms and stereotypes.

6.7.2 Operational Controls

Three operation mode buttons would be available to control the mode of the robot:

- OPERATOR button
- PLANNER – STATIONARY button
- PLANNER – MOBILE button

The rationale for these buttons was to maintain active, manual involvement for the operator during robot operations and to provide for fail-safe transitions between operation modes. The normal operator procedures would be as follows:

1. The operator presses the OPERATOR button and then uses the joystick to position the robot to the desired starting location.
2. Once the robot is at the desired location, the operator presses the PLANNER – STATIONARY button to indicate that the Planner can start executing the programmed tile tasks while stationary at the current location.

3. After the programmed, stationary tasks for the current location have been completed, the operator presses the PLANNER – MOBILE button to indicate to the Planner that MAPS move the robot to the designated position for the next set of programmed tile tasks.
4. Repeat from Step 2 until all programmed tile tasks have been completed.
5. Finally, the operator presses the OPERATOR button and switches back to using the joystick to maneuver the robot to the desired final destination.

6.7.3 System State Indicators

Two mechanical status indicators would provide information regarding the current positions of the arm and legs based on feedback from sensors. The arm status indicator would display the arm in either the stowed position or the extended position. The leg status indicator would display the positions of both legs, either in the extended or retracted positions. Software interlocks would be implemented to detect and provide warnings for potentially hazardous combinations of arm and leg movements.

Hardware interlocks have been designed on both the arm and the legs. The power supply for the legs would go through a contact switch on the arm. Since the arm should be extended only if the legs are down, the contact switch has been designed to close when the arm is stowed. Therefore, the legs could be raised only when the arm is down (switch would be closed). The power supply for the arm would go through the contact switches

on the legs; thus, if the legs were up, the contact would be broken and the arm could not be extended.

Dials were designed into the control panel to communicate the physical representation of the arm and legs to the operator. Because man can easily detect change in patterns, the operator would have the means to detect error scenarios if the arm falls or the legs change position.

In the case of emergency situations, the operator would be warned of any discrepancies found between the states of the software and hardware interlocks by a distinguishable alert and a warning message on the control panel. The warning message would attempt to explain the root cause of the problem and provide a recommended action to resolve the problem (e.g., hit the emergency stop button). This design accounted for allowing the proper amount of time for the operator to react and take the proper steps to reach problem resolution. The operator could view a log of the tasks accomplished and review pertinent historical data.

6.7.4 Training

Training sessions would be crucial to the proper usage of this human-machine interface. Since components of the control panel have been explicitly designed for active operator interaction, training sessions would focus on the subtleties of the operating procedures, overall MAPS system functions, design rationale for user interface components, and potential hazards that could arise during operations. The safety aspects of the HMI design

and general strategies covering various hazard scenarios would be addressed. Hands-on tests under normal and hazardous conditions would be part of a certification program in order for operators to be allowed to utilize this human-machine interface.

6.7.5 Human-centered Approach

The design of the HMI must take a human-centered approach to ensure safe operations. The HMI designed by another team given the same set of MAPS requirements resulted in the design in Appendix F. The HMI in Appendix F is far more complex and confusing than the HMI presented in Figure 6-1. The lesson learned is that successful HMI design must include a human-centered approach in defining intuitive, understandable interfaces with low complexity and sufficient amount of feedback such that the operator can perform his/her tasks effectively and confidently.

Chapter 7

Conclusions

7.1 Thesis Conclusions

This thesis presented a new systemic model-based framework to assess and reduce risk in human-computer controlled systems. In order to understand the basis for accidents in today's complex systems, this thesis initially investigated the root causes of recent accidents such as the Ariane 5 launcher explosion, the Titan IVB Centaur failure, the Mars Climate Orbiter mishap, and the Mars Polar Lander mission failure.

As the significance of the root causality of accidents was reinforced, traditional risk assessment approaches were studied to survey the existing methodologies and techniques with respect to analyzing risk factors and estimating risk. Cost-benefit analysis, decision analysis, probabilistic risk assessment, and human reliability analysis were among the formal mechanisms examined that aid the decision making process for risk evaluation and risk aversion.

In order to understand accidents that have occurred, several classic accident models were explored and evaluated in terms of their underlying theories and ultimately their applicability to today's complex human-computer controlled systems. In estimating future risk, most of the traditional accident models presented focused on individual component failure events/conditions as the causality of accidents. However, the systems models provided a more comprehensive approach by focusing more on the perspective of

component interactions and system processes contributing to system execution as a whole rather than isolating failures within components.

A new holistic systems accident model and framework were proposed to improve one's ability to assess and reduce risk in the design, development, and operations of complex human-computer systems. The impetus for this model-based framework was to identify a comprehensive set of risk factors early in the system design process by considering systemic influences, system attributes, and system processes.

By integrating this framework's top-down Preliminary Hazard Analysis methodologies in the system design process, one can design safety into the entire system. The new holistic system accident model-based framework concentrates on solidifying the upstream processes to maximize benefits downstream.

The Fault Tree Analysis step in this framework facilitates functional decomposition for system function hazards within a complex system. This technique allows one to divide complex problems into simpler, more manageable problems so that these simpler problems can be tackled in a focused manner to eventually reveal root causal factors. Through the new view of system accidents, one can comprehensively consider multiple hierarchical levels in determining accident causality.

Another key aspect of this framework is in the identification of design constraints and mitigation features for improved risk management. Establishing the appropriate system

bounds for system function hazards and the corresponding mitigation strategy is an essential upstream process. Rasmussen shares this systemic view pertaining to the significance of constraint identification in managing risk when he states:

> "The most promising general approach to improved risk management appears to be an *explicit identification of the operational constraints of the work space, efforts to make these constraints – the boundaries – visible to the actors and to give them opportunity to learn to cope with these boundaries.* In addition to improved safety, making boundaries visible may also increase system effectiveness in that operation close to known boundaries may be safer than requiring excessive margins which are likely to deteriorate under pressure." [29]

The final step in this framework is to design the human-machine interface to produce the system form. The manner in which the human-machine interface is designed has a significant impact on the functionality, usability, integrability, complexity, and scalability of a system. System elements are connected at the interfaces to achieve functionality that is greater than the sum of the parts. Well-defined, human-centered interfaces will lead to lower system complexity and reduced risk for accidents. System performance and safety will be maximized through system and subsystem interfaces developed holistically through the new view of system accidents.

The MAPS case study showed how this system accident model-based framework for risk assessment could be applied to a real world system. The identification and evaluation of

MAPS system function hazards, root causes, design constraints, and mitigation factors, along with a viable MAPS human-machine interface, substantiate the benefits of assimilating this framework into the upstream system design processes.

Rapid growth in technology today is altering the way in which complex systems must be designed and operated. Leveson states that "new technology and new applications of existing technology often introduce 'unknown unknowns' (sometimes referred to as UNK-UNKS)" [2]. The systems model-based framework proposed in this thesis will reduce the risk of known as well as unknown accident factors in complex systems by adhering to systemic design principles and processes. With the holistic systems accident model, the framework will identify root causal factors and detect potentially detrimental aspects in design decisions and assumptions, organizational culture, management policies, regulations, technology strategy, and system development processes before they lead to accidents. The knowledge gained from the upstream influences and drivers is infused throughout the system design process and is reflected in the resultant system.

Capitalizing on upstream wisdom and leveraging holistic systems thinking are critical ingredients in the advancement of risk assessment and risk reduction methodologies to address the mounting challenges of complex human-computer controlled systems that exist today and that will exist in the future. As theoretical physicist David Bohm once aptly stated:

> "Man's general way of thinking of the totality ... is crucial for overall order of the human mind itself. If he thinks of the totality as constituted of independent

fragments then that is how his mind will tend to operate, but if he can include everything coherently and harmoniously in an overall whole that is undivided, unbroken ... then his mind will tend to move in a similar way, and from this will flow an orderly action within the whole." [30]

7.2 Future Research

This thesis contributed the initial phase of research in establishing a new, model-based framework for risk assessment in human-computer controlled systems. Further phases would entail the incorporation of additional cognitive human error model methodologies and further reflection of modern organizational theory about accidents. Other considerations would encompass multiple cognitive models (for a complex system with multiple control points) and devising new systemic hazard analysis approaches within the framework.

Another phase would be an in-depth demonstration and evaluation of the new framework. One approach for evaluating the framework would involve the application to accidents of which causality has been determined. An analysis would be made between the causal factors determined from the accident and the causal factors resulting from applying the framework.

A subsequent phase would involve the experimental application to other real systems. Further case studies would apply the framework to appropriate real systems and assess the findings for framework validation and enhancement.

Appendix A

Ariane 5 Analysis of Failure: Chain of Technical Events

Based on the extensive documentation and data on the Ariane 501 failure made available to the Board, the following chain of events, their inter-relations and causes have been established, starting with the destruction of the launcher and tracing back in time towards the primary cause.

- The launcher started to disintegrate at about H0 + 39 seconds because of high aerodynamic loads due to an angle of attack of more than 20 degrees that led to separation of the boosters from the main stage, in turn triggering the self-destruct system of the launcher.
- This angle of attack was caused by full nozzle deflections of the solid boosters and the Vulcain main engine.
- These nozzle deflections were commanded by the On-Board Computer (OBC) software on the basis of data transmitted by the active Inertial Reference System (SRI 2). Part of these data at that time did not contain proper flight data, but showed a diagnostic bit pattern of the computer of the SRI 2, which was interpreted as flight data.
- The reason why the active SRI 2 did not send correct attitude data was that the unit had declared a failure due to a software exception.
- The OBC could not switch to the back-up SRI 1 because that unit had already ceased to function during the previous data cycle (72 milliseconds period) for the same reason as SRI 2.
- The internal SRI software exception was caused during execution of a data conversion from 64-bit floating point to 16-bit signed integer value. The floating point number which was converted had a value greater than what could be represented by a 16-bit signed integer. This resulted in an Operand Error. The data conversion instructions (in Ada code) were not protected from causing an Operand Error, although other conversions of comparable variables in the same place in the code were protected.
- The error occurred in a part of the software that only performs alignment of the strap-down inertial platform. This software module computes meaningful results only before lift-off. As soon as the launcher lifts off, this function serves no purpose.
- The alignment function is operative for 50 seconds after starting of the Flight Mode of the SRIs which occurs at H0 - 3 seconds for Ariane 5. Consequently, when lift-off occurs, the function continues for approx. 40 seconds of flight. This time sequence is based on a requirement of Ariane 4 and is not required for Ariane 5.
- The Operand Error occurred due to an unexpected high value of an internal alignment function result called BH, Horizontal Bias, related to the horizontal velocity sensed by the platform. This value is calculated as an indicator for alignment precision over time.

- The value of BH was much higher than expected because the early part of the trajectory of Ariane 5 differs from that of Ariane 4 and results in considerably higher horizontal velocity values.

The SRI internal events that led to the failure have been reproduced by simulation calculations. Furthermore, both SRIs were recovered during the Board's investigation and the failure context was precisely determined from memory readouts. In addition, the Board has examined the software code which was shown to be consistent with the failure scenario. The results of these examinations are documented in the Technical Report.

Therefore, it is established beyond reasonable doubt that the chain of events set out above reflects the technical causes of the failure of Ariane 501.

Source: [3]

Appendix B

Mars Climate Orbiter Analysis of Failure: Root Cause and Contributing Causes

The Board recognizes that mistakes occur on spacecraft projects. However, sufficient processes are usually in place on projects to catch these mistakes before they become critical to mission success. Unfortunately for MCO, the root cause was not caught by the processes in-place in the MCO project.

A summary of findings, contributing causes and MPL recommendations are listed below. These are described in more detail in the body of this report along with the MCO and MPL observations and recommendations.

Root Cause: Failure to use metric units in the coding of a ground software file, "Small Forces," used in trajectory models

Contributing Causes:

1. Undetected mismodeling of spacecraft velocity changes
2. Navigation Team unfamiliar with spacecraft
3. Trajectory correction maneuver number 5 not performed
4. System engineering process did not adequately address transition from development to operations
5. Inadequate communications between project elements
6. Inadequate operations Navigation Team staffing
7. Inadequate training
8. Verification and validation process did not adequately address ground software

MPL Recommendations:

- Verify the consistent use of units throughout the MPL spacecraft design and operations
- Conduct software audit for specification compliance on all data transferred between JPL and Lockheed Martin Astronautics
- Verify Small Forces models used for MPL
- Compare prime MPL navigation projections with projections by alternate navigation methods
- Train Navigation Team in spacecraft design and operations
- Prepare for possibility of executing trajectory correction maneuver number 5
- Establish MPL systems organization to concentrate on trajectory correction maneuver number 5 and entry, descent and landing operations
- Take steps to improve communications
- Augment Operations Team staff with experienced people to support entry, descent and landing

- Train entire MPL Team and encourage use of Incident, Surprise, Anomaly process
- Develop and execute systems verification matrix for all requirements
- Conduct independent reviews on all mission critical events
- Construct a fault tree analysis for remainder of MPL mission
- Assign overall Mission Manager
- Perform thermal analysis of thrusters feedline heaters and consider use of pre-conditioning pulses
- Reexamine propulsion subsystem operations during entry, descent, and landing

Source: [6]

Appendix C

Mobility and Positioning Software (MAPS)

Description:
MAPS (Mobility and Positioning Software) is part of a tessellator robot system designed to service tiles (the thermal protection system) on the Space Shuttle[1].

The Orbiter is covered with several types of heat resistant tiles that protect the orbiter's aluminum skin during the heat of reentry. While the majority of the upper surfaces are covered with flexible insulation blankets, the lower surfaces are covered with silica tiles. These tiles have a glazed coating over soft and highly porous silica fibers. The tiles are 95% air by volume, which makes them extremely light but also makes them capable of absorbing a tremendous amount of water. Water in the tiles causes a substantial weight problem that can adversely affect launch and orbit capabilities for the shuttles. Because the orbiters may be exposed to rain during transport and on the launch pad, the tiles must be waterproofed. This task is accomplished through the use of a specialized hydrophobic chemical, DMES, which is injected into each and every tile by the robot. There are approximately 17,000 lower surface tiles covering an area that is roughly 25m x 40m.

The tessellator robot also inspects the tiles. During launch, reentry, and transport, a number of defects can occur on the tiles. These defects are evident as scratches, cracks, gouges, discoloring, and erosion of surfaces. The tiles are examined for such defects to determine if they warrant replacement, repair, or no action. The typical procedure involves visual inspection of each tile to see if there is any damage and then assessment and categorization of the defects according to detailed checklists. Later, work orders are issued for repair of individual tiles.

The robot inspects each tile and injects a toxic waterproofing chemical, which prevents the lightweight, silica tiles from absorbing water. Because there are so many tiles, Tessellator divides or tessellates, its work area among uniform work spaces, inspecting tiles in each area with as little overlap between work spaces as possible.

Before each inspection shift, a supervisor enters instructions into Tessellator about shuttle position and inspection sequence via an off-board computer, the Workcell Controller. Tessellator then uses a rotating laser to position itself under the shuttle; the robot's camera locates the exact tile to be inspected. Because the shuttle's belly is not flat, Tessellator customizes its upward movement to each tile: Two vertical beams on either side of the robot raise the manipulator arm, which holds the injection tools and camera; a smaller lifting device raises the arm the rest of the way.

---

[1] The robot was designed as a research project in the Robotics Dept. at CMU. This specification was derived from one that students pursuing a master's degree in CS created for a project at the SEI. Changes have been made from the original specification in order to satisfy different goals.

By comparing the current state of each tile with the state of the tile at previous inspections, Tessellator characterizes anomalies in tiles as cracks, scratches, gouges, discoloring, or erosion. The robot also indicates when it is unsure what is wrong with a tile, so the supervisor can reanalyze the tile on the screen of the Workcell Controller. At the end of a shift, Tessellator's updated tile information is entered into existing NASA databases.

On board, a computer controls Tessellator's high-level processing tasks while a low-level controller and amplifiers direct arm and wheel motions. Two more computers control the robot's vision and injection systems. If anything goes wrong – rising compartment temperatures, low battery level, or other changes – safety circuits will shut the robot down, and Tessellator will correct the problem.

MAPS (the mobility and positioning software) issues movement commands to the motor controller, which controls the mobile base of the robot. MAPS in turn is controlled either by the operator or an on-board computer called the Planner. The operator controls robot movement and positioning by providing MAPS with a specification of the destination and route.

The tessellator robot is unstable when the manipulator arm is extended, so stabilizer legs are used to provide stability. These legs must be retracted when the robot is in motion. MAPS is responsible for controlling the stabilizer legs.

At the beginning of a shift, the Tessellator is downloaded a job. The job consists of a series of files describing tile locations, sequences, target Ids, orbiter parking measurements, etc. The job is created on the Workcell Controller, an off-board workstation that is used to create jobs and update other NASA databases after the robot uploads data gathered during the course of the shift. This data includes tile images, records of tiles injected or inspected, and other pertinent job data. In addition, robot status data is used to monitor robot operation.

Environment Issues:
The work areas can be very crowded. The robot must negotiate jackstands, columns, workstands, cables, and hoses. In addition, there are hanging cords, clamps, and hoses. Because the system might cause damage to the ground obstacles, cable covers are used for protection and the robot system must traverse these covers.

Source: [26]

Appendix D

MAPS Fault Tree Analysis

Figure D-1 to Figure D-4 are the work products from the Fault Tree Analysis performed on MAPS hazard events.

Source: [27]

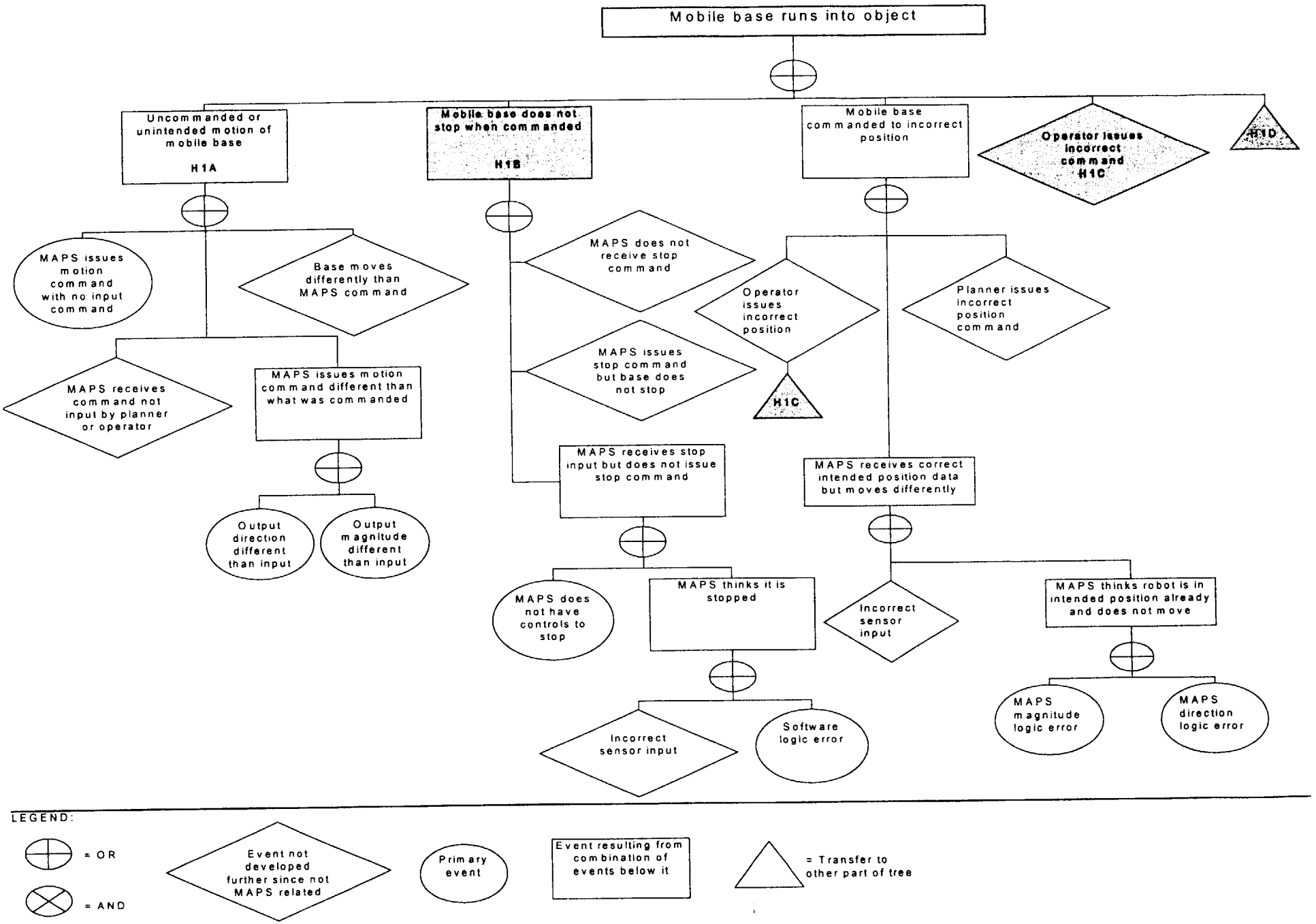


Figure D-1, Part 1: MAPS Fault Tree Analysis

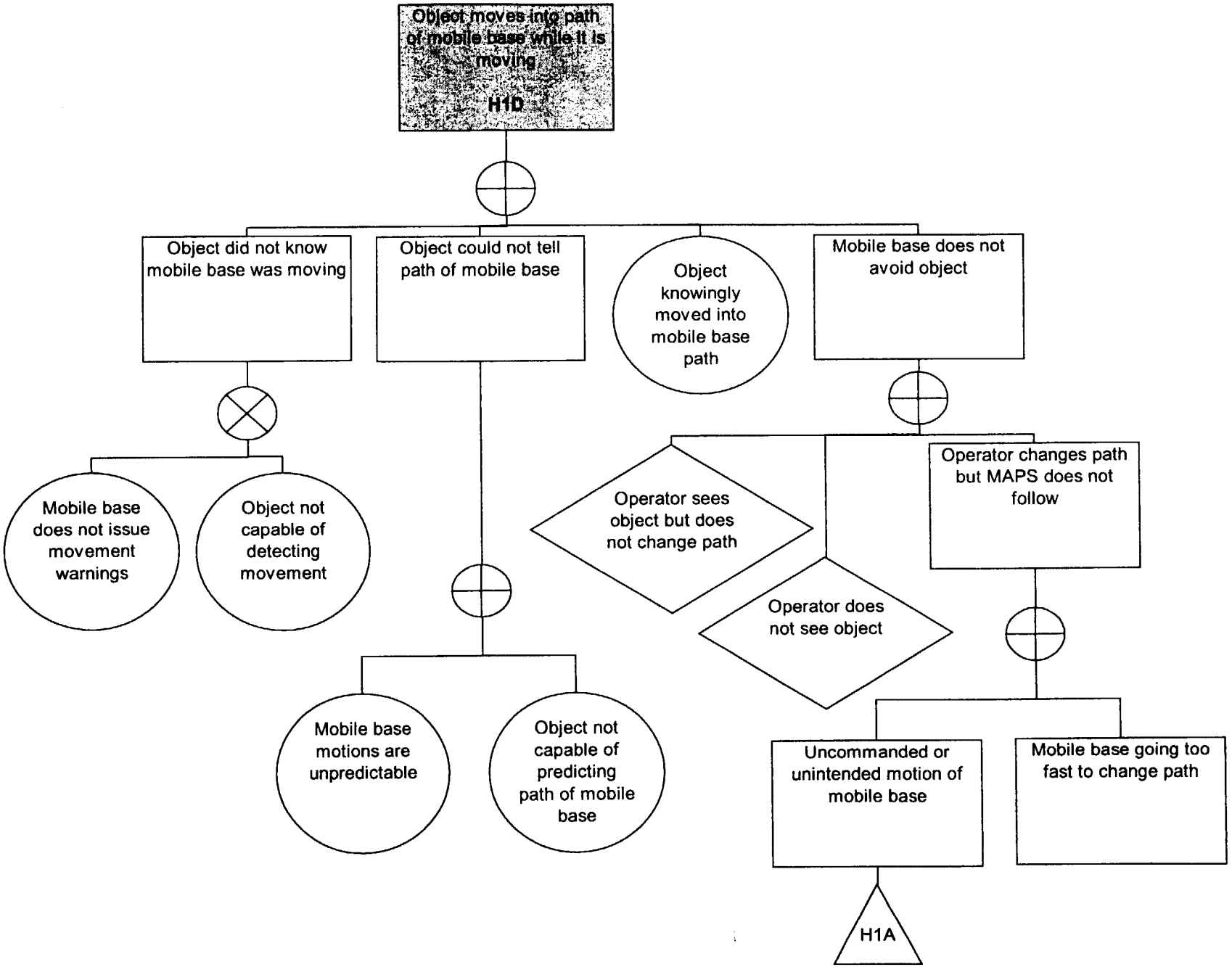


Figure D-1, Part 2: MAPS Fault Tree Analysis

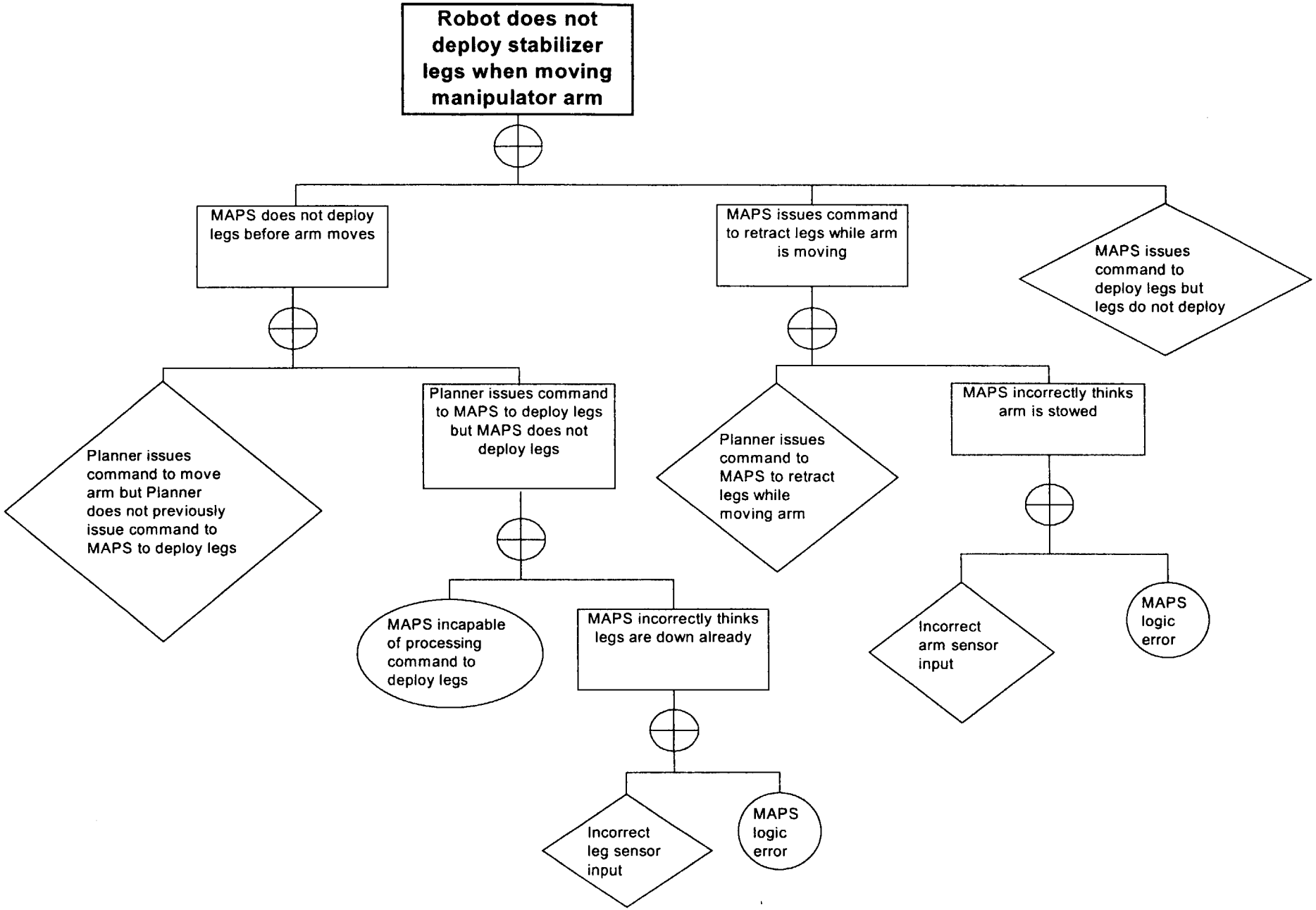


Figure D-2: MAPS Fault Tree Analysis

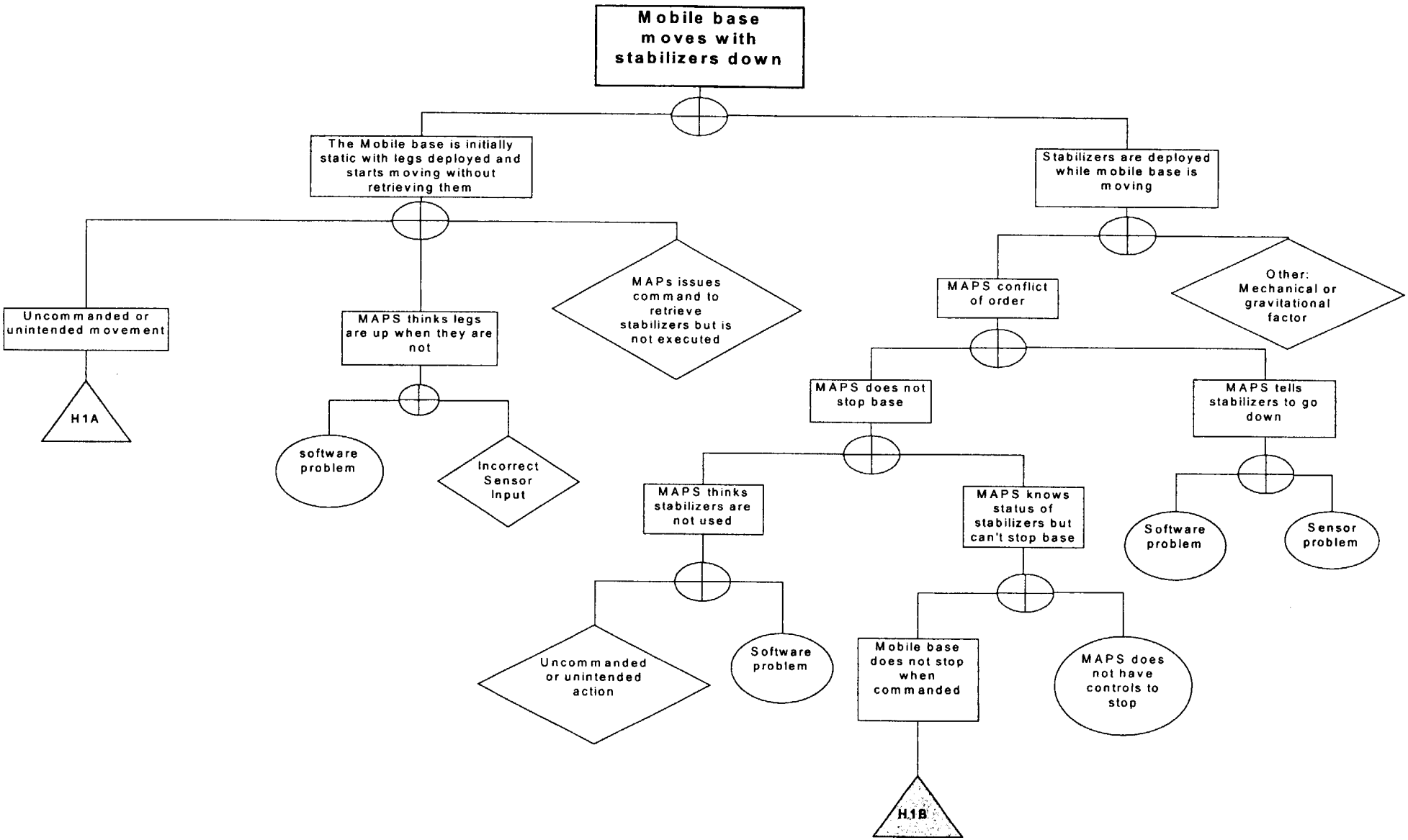


Figure D-3: MAPS Fault Tree Analysis

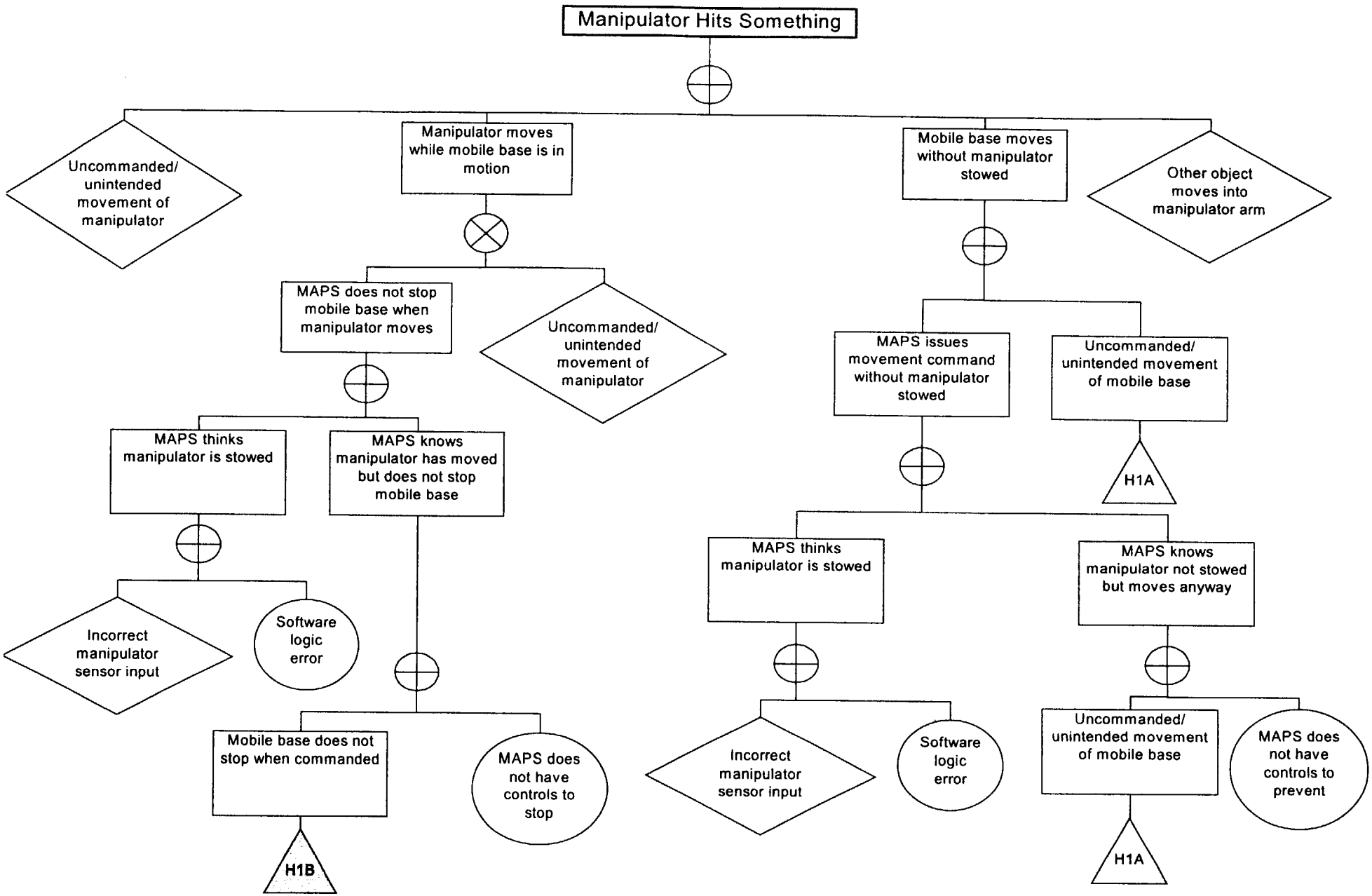


Figure D-4: MAPS Fault Tree Analysis

Appendix E

MAPS Design Constraints and Mitigation Features

Figures E-1 to Figure E-5 are the work products from the design constraint and mitigation feature assessment for MAPS hazard events. Source: [27]

Constraint #	Safety Design Constraint	Mitigation Feature	Notes
1.	Mobile base shall not run into objects	See next levels down	Too general at this level to have a mitigation feature
1.1.	Mobile base shall move only when commanded	Mobile base shall require two steps or two inputs before moving - e.g. must "arm" first before moving	Should investigate human factors for operation of base and trade-off between ease of operation and improved safety
1.1.1.	System shall be protected from electromagnetic interference	All wires must be shielded. All controllers to have EMI protection.	This is not a MAPS requirement
1.1.2.	MAPS shall issue output only in response to an input	MAPS must check for outputs issued with no input	Software check
1.1.2.1.	MAPS output shall be in same direction as input command	MAPS must check for same polarity between input signal and output signal	Software check
1.1.2.2.	MAPS output shall have magnitude in proportion to input command	MAPS must check for output magnitude in proportion to input	Software check
1.1.3.	MAPS shall monitor base for correct movement	Actual base movement must be fed back to MAPS for comparison with commanded movement	Assumes that base has some method of measuring movement to feed back to MAPS
1.1.4.	MAPS must receive correct input signals	Redundant input signals shall be used in order to confirm signals are correct; assertions shall be used for expected values of inputs	See also general note at bottom about system level checks
1.1.4.1.	MAPS shall detect invalid input signals	MAPS shall compare redundant signals for agreement	Software check
1.1.4.2.	MAPS shall stop mobile base if invalid input signals are found	If input signals do not agree, MAPS shall stop mobile base	Internal to MAPS

Figure E-1, Part 1: MAPS Design Constraints and Mitigation Features

1.2.	Mobile base shall stop when commanded	Use of proper assertions in programming that will allow to check for correct input by fail-safe processing modules. In this regard, sensors must be checked prior to operation, check for specific required type of input, determine what are the desired or undesired type of output, specified limit, abnormal value. Mobile base shall have passive braking system in event active braking does not work (like the brakes on power tools that stop the blades from spinning when the tool is turned off)	See also general note at bottom about system level checks
1.2.1.	MAPS must receive stop command	Redundant input signals shall be used in order to confirm signals are correct; assertions shall be used for expected values of inputs	See also general note at bottom about system level checks
1.2.1.1.	MAPS shall detect invalid stop command	This is part of the initialization routine for the sensor /software control issues.	An invalid stop command will be detected by comparing current values for initialization with the defined parameters assigned to the sensors
1.2.1.2.	MAPS shall stop mobile base if invalid stop command is found	An abnormal value will be caught by the assertions routines created as part of the software control system.	Limits in the acceleration, velocity, and position are provided as part of the design specification and requirements. A comparison with all this data will help detect invalid commands and undesired movements, providing at the same time with proper alerts.
1.2.2.	Mobile base must receive stop command from MAPS	Looping period must be established for updating of signals to/from MAPS. Limiting or setting a limit for long response or short response will help determine if the Mobile Base received a signal or not.	See general note at bottom about system architecture and checks
1.2.3.	MAPS shall issue stop command when it receives stop input	A feedback loop could be created so that when the error propagates a certain threshold, then MAPS can receive an update on the situation (being position[like no stop] or velocity [it doesn't slow down])	See general note at bottom about system architecture and checks
1.2.3.1.	MAPS must receive correct mobile base speed information	Determination of proper speed by sensor should be accomplished by reading and integrating from two independent systems. MAPS can then compare and decide to alert in case the two readings differ.	Assumes there are two independent systems available to measure speed of mobile base

Figure E-1, Part 2: MAPS Design Constraints and Mitigation Features

1.3.	Mobile base shall move to correct position	Correct input information is confirmed by the software control logic, previously outlined in section 1.2. Most of the functionality's are described there.	
1.3.1.	MAPS shall receive correct position information	Determination of proper speed by sensor should be accomplished by reading and integrating from two independent systems. MAPS can then compare and decide to alert in case the two readings differ.	Similar to 1.2.3.1
1.3.2.	MAPs shall issue output command in accordance with input	Feedback control loop must be set as part of the assertion logic that will measure the threshold of error present in the control system.	Software control
1.3.2.1.	Output shall be in direction of input	There are several factors to consider in this area: - human factors issues regarding type of display(self-centered) or changing orientation; - factors regarding intermediate stages of operation corresponding to regions and operations where more delicate and precise operations are needed, the software control logic will adapt to different levels of manual control as opposed to when a fast reaction is needed.	
1.3.2.2.	Output shall have magnitude in accordance with input	This relates again to point 1.3.2.1	

1.4.	MAPS shall help operator by providing status information back to operator	System will display robot position and movement information in graphical format	Need to investigate how to best display directionality (i.e., usability complications may arise when robot is heading towards operator and operator needs to change direction using joystick)

Figure E-1, Part 3: MAPS Design Constraints and Mitigation Features

1.5.	Mobile base shall avoid objects that enter its path	MAPS will stop mobile base and prevent it from further movement towards an object once detected in its path (feedback from object detection system)	Assumes object detection system that feeds information to MAPS
1.5.1.	Mobile base shall not be commanded to an occupied position	MAPS will stop mobile base and prevent it from further movement towards an object once detected in its path (feedback from object detection system)	Assumes object detection system that feeds information to MAPS
1.5.2.	MAPS shall provide movement warnings	MAPS will activate warning flashers and/or warning sounds while mobile base is moving	Assumes warning systems are available for MAPS to activate
1.5.3.	Mobile base shall have predictable motions	MAPS will detect and stop processing abnormal sequence of movement commands; MAPS will implement assertions to calculate closest acceptable trajectory/path to recorrect movement.	In addition, design of base could prevent sudden motions - i.e. motor torque capability relative to inertia of base
1.5.3.1.	MAPS shall not make abrupt changes in direction	MAPS will detect and stop processing abrupt changes in direction; MAPS will implement assertions to calculate closest acceptable trajectory/path to recorrect movement.	In addition, design of base could prevent sudden motions - i.e. motor torque capability relative to inertia of base
1.5.3.2.	MAPS shall not make any sudden accelerations	MAPS will detect and stop processing sudden accelerations; MAPS will implement assertions to calculate closest acceptable velocity/acceleration change.	In addition, design of base could prevent sudden motions - i.e. motor torque capability relative to inertia of base
1.5.4.	MAPS shall limit speed of mobile base	Output signal from MAPS will be subject to hardware saturation; MAPS will implement assertions for acceptable values on the output signal from MA+C9PS; Feedback loop will poll for speed.	Using only hardware limit allows possibility of hazard if software is reused on system without hardware limit.
1.5.5.	System shall detect other objects and provide information to operator	System will detect objects in path (feedback from object detection system) and display detection information in graphical format	Assumes object detection system that feeds information to MAPS

Figure E-1, Part 4: MAPS Design Constraints and Mitigation Features

2.	Manipulator arm shall move only after stabilizer legs have been deployed	System will implement hardware interlock between legs and manipulator arm. Power supply for arm goes through contacts on legs. If legs are up, contact is broken and arm cannot move.	See notes at back on fail-safe scenarios - decision made to have legs fail in the up position
2.1.	MAPS shall deploy legs before arm moves	MAPS will receive command to deploy legs before arm is moved	Software control added to work in conjunction with hardware interlock
2.1.1.	MAPS shall receive correct sensor input on status of stabilizer legs	MAPS will implement assertions for expected values of inputs; MAPS will propogate sensor input into overall system communications loop	
2.1.1.1.	MAPS shall detect invalid sensor input	MAPS will implement assertions to validate sensor input for stablizer legs	Software check
2.1.1.2.	MAPS shall provide status of legs to arm controller	MAPS will maintain state model for leg status; MAPS will propogate leg status into overall system communications loop	Assumes communication between MAPS and planner or some other controller for arm
2.2.	MAPS shall not retract legs while arm is moving	MAPS will verify arm is in stowed state before issuing command to retract legs	Use contact switch in 2.2.1.1
2.2.1.	MAPS shall receive correct sensor input on status of arm	MAPS will implement assertions for expected values of inputs; MAPS will propogate sensor input into overall system communications loop	Software check
2.2.1.1.	MAPS shall not retract legs if arm is not in stowed position	System will implement contact switch that is closed when arm is in stowed position. Legs can only be raised when contact switch is closed.	Combination hardware/software
2.3.	Stabilizer legs must receive commands from MAPS	MAPS will be the only user of the stabilizer legs interface; MAPS will propogate leg status into overall system communications loop	

Figure E-2: MAPS Design Constraints and Mitigation Features

3.	Mobile base shall move only when stabilizer legs are up	Sensors should be able to perform to specifications. Please refer to sensor initialization routines for safety requirements in the sensor system. Check also for the mechanical interlock (3.1)	
3.1.	Mobile base shall not move before legs are raised	Putting legs down opens contacts for circuit providing power to the mobile base. Lifting legs closes circuit.	Mechanical interlock independent of software control system.
3.1.1.	MAPS shall receive correct information on status of legs	Statements about the expected values of the legs are issued from a looping sequence where data is obtained at different intervals.	This will allow us to do range checks over the input values and make comparisons
3.1.2.	Stabilizer legs must receive commands from MAPS	S/A above. Also include the set of limits outlined in what respect to position and velocity. See notes for an explanation.	We can state with a degree of certainty that the mobile base will not escape a set of dimensions, that it will not exceed a specific speed, or that it will not accelerate to a specific limit.
3.2.	MAPS shall not deploy stabilizer legs while mobile base is moving	Software control so that if mobile base is moving, legs can not be deployed.	Correct settings should be enabled for the detection of stabilizers while in use
3.2.1.	MAPS shall stop base if legs deploy	S/A above	Restrictions should be place for unintended actions during mode of operation
3.3	MAPS should be able to respond safely to a conflict of order	Different modes of operation and levels of controls should be tested to completion to prevent conflicts in control.	

Figure E-3: MAPS Design Constraints and Mitigation Features

4.	Manipulator shall not hit other objects	See next levels down	Too general at this level to have a mitigation feature
4.1.	Manipulator shall move only when commanded	Planner must compare output for manipulator with input for consistency	Not a MAPS requirement
4.2.	Manipulator shall move only when base is stopped	Interlock shall be used between mobile base movement and manipulator movement - if base is moving, manipulator can not - if manipulator is not stowed, base can not move	For all of these, think about if there are cases when you WOULD want to do these things - emergencies for example
4.2.1.	MAPS shall stop mobile base if manipulator moves	S/A above - will not allow manipulator to move if base is moving	Use contact switch as described above
4.2.2.	MAPS shall receive correct information on status of manipulator	Redundant signals for status of manipulator shall be sent to MAPS	Decide on best way to do this
4.3.	Mobile base shall not move without manipulator stowed	S/A above - will not allow base to move if manipulator is not in stowed position	Use contact switch as described above
4.4.	System shall detect object moving into path of manipulator arm	No action - decision made that this would be too complicated to do in comparison with the potential benefit	

Figure E-4: MAPS Design Constraints and Mitigation Features

System level checks	At the system level, use a type of time-triggered architecture along with checks that will be made. Each subsystem is expected to communicate its status at a given time interval. If that subsystem does not communicate then it is assumed to have failed. In addition, each individual subsystem will conduct internal checks to make sure it is working properly. This will assure the proper operation of things like sensors, etc. So if the components are being checked and the individual subsystems are communicating their status that things are OK, then this system level approach should take care of many of the above mitigation requirements for communication of information between subsystems.

Fail-safe Scenario 1	If the legs fail in the up position (they are spring loaded to return to the up position if they lose power) this will allow the base to move. However, if the arm is extended when this happens, we have said that the legs cannot raise unless the arm is stowed, so this is a conflict. If raising the legs cuts power to the arm, then the arm can not move, but since the base cannot move if the arm is not stowed, then the robot will be paralized if this happens.
Fail-safe Scenario 2	If the legs fail in the down position (they go down by gravity if they lose power) this will cause the mobile base to stop. If the base is stopped, it will remain stopped. If the base is moving, you would like it to stop before the legs actually touch the ground, or else there will be damage. Assuming you can do this by damping the lowering of the arms, the base will be stopped. If the legs were up to begin with, the arm must have been stowed. With the base stopped and the legs down, the arm can be used, but this probably does not help you any.
Note	In the event of a failure, the thing you would most likely want to be able to do is get the robot to a location where it can be repaired. Neither of these scenarios would allow that, but with the legs down, you certainly can not drive the robot. With the legs up, if the arm is stowed you could drive the robot. If the arm is not stowed, the way to address this is to have the natural state of the arm be the stowed position, so that it returns to the stowed position if it loses power.
Note	The problem with having the legs fail in the up position, is that if the arm is in use at the time when they fail, raising them may allow the robot to tip over. However, if they fail down, but only under the power of gravity, this also may not be enough to prevent the robot from tipping over. As such, there is not benefit either way unless the legs are actively held down, which would be hard to do if there is no power. This is somewhat irrelevant if the arm automatically returns to the stowed position as described above.
Conclusion	We chose to have the legs fail in the up position. Based on the above line of reasoning, this seems to be the best situation given that the arm returns to the stowed position if it loses power. In this case, the base will be able to be driven to be repaired.

General Note	For most of the above hardware controls, the same controls should be implemented in software as well for two reasons: this provides redundancy in case the hardware control fails, and this keeps the functionality in case the software is reused in a system that does not have the same hardware controls (e.g. Therac 25 or Ariane 5) - the designer has some responsibility to account for foreseeable misuse

General Note	For all of the hardware interlocks (software too, although software can change performance under a failed condition), need to think about if there are cases when you WOULD want to do the things that are being prevented by the interlocks- emergencies for example - e.g. is there some set of circumstances when you would want to be able to move the arm when the legs are up? We did not think of any, but we certainly did not do an exhaustive study

Figure E-5: MAPS Design Constraints and Mitigation Feature

Appendix F

MAPS HMI Example with Excessively High Complexity

Source: [28]

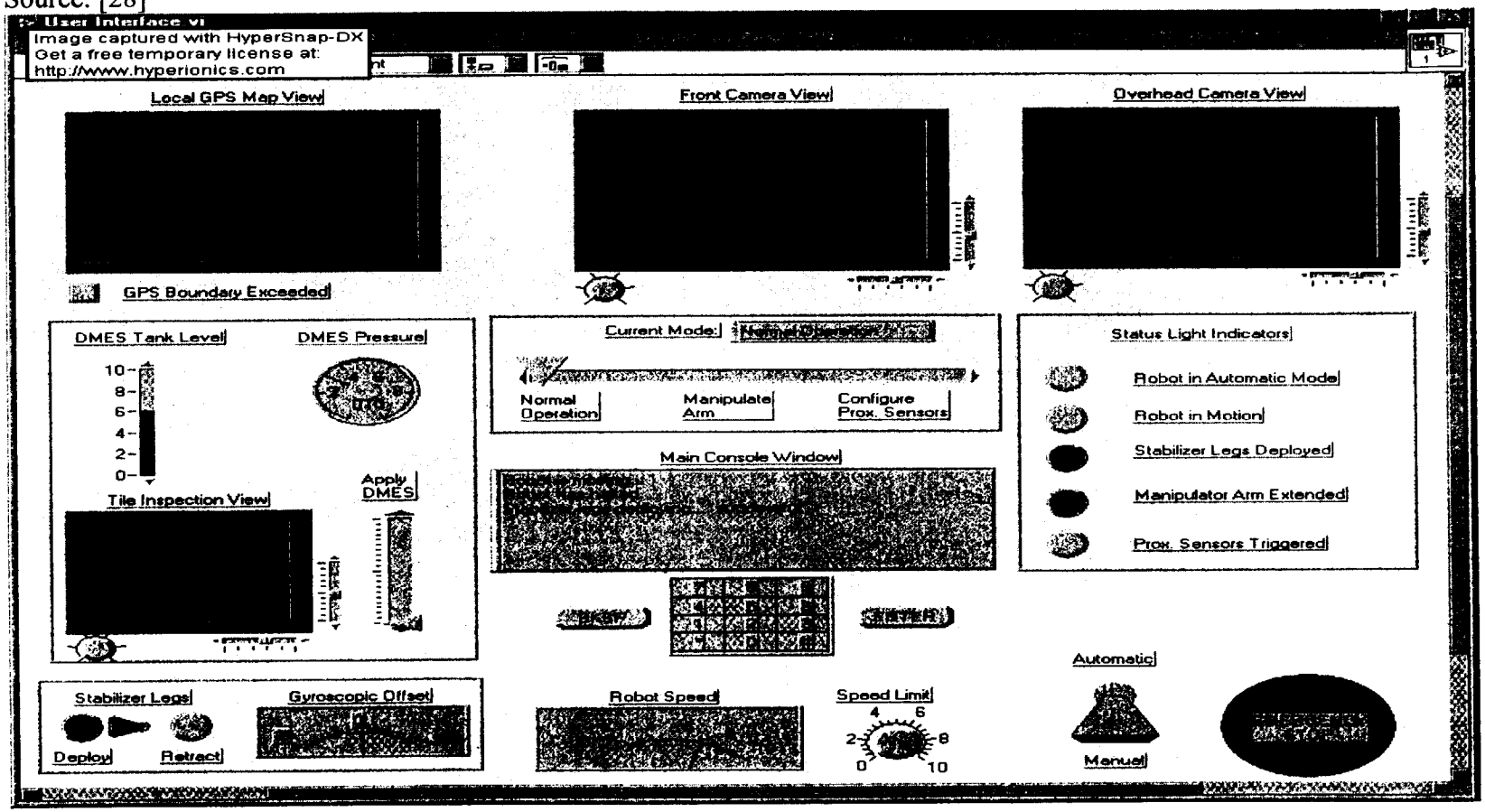


Figure F-1: HMI Design with Excessively High Complexity

Bibliography

1. Billings, Charles E., *Aviation Automation: The Search for a Human-Centered Approach*, Lawrence Erlbaum Associates, Inc., Mahwah, NJ, 1997.

2. Leveson, Nancy G., *Safeware: System Safety and Computers*, Addison-Wesley, Reading, MA, 1995.

3. Lions, Jacques-Louis, *Ariane 5 Flight 501 Failure: Report by the Inquiry Board*, Paris, 1996.

4. Schefter, Jim, "NASA's Changing Fortunes," *Popular Science*, March 17, 2000.

5. Director, Operational Test & Evaluation, *FY99 Annual Report*, Washington, D.C., 1999.

6. National Aeronautics and Space Administration, *Mars Climate Orbiter Mishap Investigation Board: Phase I Report*, Washington, D.C., 1999.

7. Jet Propulsion Laboratory Special Review Board, *Report on the Loss of the Mars Polar Lander and Deep Space 2 Missions*, Pasadena, CA, 2000.

8. Fischhoff, Baruch, Sarah Lichtenstein, Paul Slovic, Stephen L. Derby, and Ralph L. Keeney, *Acceptable Risk*, Cambridge University Press, New York, NY, 1981.

9. Singleton, William Thomas, and Jan Hovden, *Risk and Decisions*, John Wiley and Sons, New York, NY, 1987.

10. Rasmussen, Jens, *NATO Symposium on Human Detection and Diagnosis of System Failures*, Plenum Press, New York, NY, 1981.

11. Apostolakis, George E., P. Kafka, and G. Mancini, *International Seminar on Accident Sequence Modeling: Human Actions, System Response, Intelligent Decision Support*, Elsevier Applied Science Publishers Ltd., New York, NY, 1988.

12. Rechtin, Eberhardt, and Mark W. Maier, *The Art of Systems Architecting*, CTC Press LLC, 1997.

13. Leveson, Nancy, and John Hansman, *Center Software Initiative Proposal (CSIP) for the NASA Software IV & V Facility*, Cambridge, MA, 1999.

14. Fischhoff, Baruch, Christoph Hohenemser, Roger Kasperson, and Robert Kates, *Risk and Chance*, Open University Press, Milton Keynes, United Kingdom, 1980.

15. Swain, Alan, and H. Guttmann, *Handbook of Human Reliability Analysis with Emphasis on Nuclear Power Plant Applications*, Sandia National Laboratories, Albuquerque, NM, 1983.

16. Reason, James, *Human Error*, Cambridge University Press, New York, NY, 1990.

17. Rasmussen, Jens, Keith Duncan, and Jacques Leplat, *New Technology and Human Error*, John Wiley & Sons Ltd., New York, NY, 1987.

18. Heinrich, Hebert W., Dan Petersen, and Nestor Roos, *Industrial Accident Prevention*, McGraw-Hill Book Company, New York, NY, 1980.

19. Petersen, Dan, *Human Error Reduction and Safety Management*, Van Nostrand Reinhold, New York, NY, 1996.

20. Rechtin, Eberhardt, and Mark W. Maier, *The Art of Systems Architecting*, CRC Press LLC, Boston, MA, 1997.

21. Rechtin, Eberhardt, *Systems Architecting: Creating and Building Complex Systems*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1991.

22. Perrow, Charles, *Normal Accidents*, Basic Books Inc., New York, NY, 1984.

23. Rasmussen, Jens, *Information Processing and Human-Machine Interaction: An Approach to Cognitive Engineering*, Elsevier Science Publishing Co., Inc., New York, NY, 1986.

24. Parasuraman, Raja, and Mustapha Mouloua, *Automation and Human Performance: Theory and Applications*, Lawrence Erlbaum Associates, Inc., Mahwah, NJ, 1996.

25. Crawley, Ed, Lectures and notes from Course 16.882: System Architecture, Massachusetts Institute of Technology, Fall 1999.

26. Leveson, Nancy G., Lectures and notes from Course 16.982: System and Software Safety, Massachusetts Institute of Technology, Spring 1999.

27. Demerly, Jon, Iwao Hatanaka, and Mario Rodriguez, Course project from Course 16.982: System and Software Safety, Massachusetts Institute of Technology, Spring 1999.

28. Hintersteiner, Jason, David McLain, Sara Pickett, Course project from Course 16.982: System and Software Safety, Massachusetts Institute of Technology, Spring 1999.

29. Rasmussen, Jens, Keynote address for Conference on Human Interaction with Complex Systems, Dayton, OH, 1996.

30. Bohm, David, *Wholeness and the Implicate Order*, Routledge and Kegan Paul, Ltd., London, England, 1996.