System Safety Engineering: Back To The Future

Nancy G. Leveson

Aeronautics and Astronautics
Massachusetts Institute of Technology

We pretend that technology, our technology, is something of a life force, a will, and a thrust of its own, on which we can blame all, with which we can explain all, and in the end by means of which we can excuse ourselves.

— T. Cuyler Young
Man in Nature

DEDICATION: To all the great engineers who taught me system safety engineering, particularly Grady Lee who believed in me, and to C.O. Miller who started us all down this path. Also to Jens Rasmussen, whose pioneering work in Europe on applying systems thinking to engineering for safety, in parallel with the system safety movement in the United States, started a revolution.

ACKNOWLEDGEMENT: The research that resulted in this book was partially supported by research grants from the NSF ITR program, the NASA Ames Design For Safety (Engineering for Complex Systems) program, the NASA Human-Centered Computing, and the NASA Langley System Architecture Program (Dave Eckhart). program.

Preface

I began my adventure in system safety after completing graduate studies in computer science and joining the faculty of a computer science department. In the first week at my new job, I received a call from Marion Moon, a system safety engineer at what was then Ground Systems Division of Hughes Aircraft Company. Apparently he had been passed between several faculty members, and I was his last hope. He told me about a new problem they were struggling with on a torpedo project, something he called software safety. I told him I didn't know anything about it, that I worked in a completely unrelated field, but I was willing to look into it. That began what has been a twenty-two year search for a solution to his problem.

It became clear rather quickly that the problem lay in system engineering. After attempting to solve it from within the computer science community, in 1998 I decided I could make more progress by moving to an aerospace engineering department, where the struggle with safety and complexity had been ongoing for a long time. I also joined what is called at MIT the Engineering Systems Division (ESD). The interactions with my colleagues in ESD encouraged me to consider engineering systems in the large, beyond simply the technical aspects of systems, and to examine the underlying foundations of the approaches we were taking. I wanted to determine if the difficulties we were having in system safety stemmed from fundamental inconsistencies between the techniques engineers were using and the new types of systems on which they were being used. I began by exploring ideas in systems theory and accident models.

Accident models form the underlying foundation for both the engineering techniques used to prevent accidents and the techniques used to assess the risk associated with using the systems we build. They explain why accidents occur, that is, the mechanisms that drive the processes leading to unacceptable losses, and they determine the approaches we take to prevent them. Most of the accident models underlying engineering today stem from the days before computers, when engineers were building much simpler systems. Engineering techniques built on these models, such as Failure Modes and Effects Analysis (FMEA) and Fault Tree Analysis (FTA), have been around for over 40 years with few changes while, at the same time, engineering has been undergoing a technological revolution. New technology is making fundamental changes in the etiology of accidents, requiring changes in the explanatory mechanisms used to understand them and in the engineering techniques applied to prevent them. For twenty years I watched engineers in industry struggling to apply the old techniques to new software-intensive systems—expending much energy and having little success—and I decided to search for something new. This book describes the results of that search and the new model of accidents and approaches to system safety engineering that resulted.

My first step was to evaluate classic chain-of-events models using recent aerospace accidents. Because too many important systemic factors did not fit into a traditional framework, I experimented with adding hierarchical levels above the basic events. Although the hierarchies helped alleviate some of the limitations, they were still unsatisfactory [67]. I concluded that we needed ways to achieve a more complete and less subjective understanding of why particular accidents occurred and how to prevent future ones. The new models also had to account for the changes in the accident mechanisms we are starting to experience in software-intensive and other high-tech systems. I decided that these goals could be achieved by building on the systems approach that was being applied by Jens Rasmussen and his followers in the field of human–computer interaction.

The ideas behind my new accident model are not new, only the way they are applied. They stem from basic concepts of systems theory, the theoretical underpinning of systems engineering as it developed after World War II. The approach to safety contained in this book is firmly rooted in systems engineering ideas and approaches. It also underscores and justifies the unique approach to engineering for safety, called System Safety, pioneered in the 1950s by aerospace engineers like C.O. Miller, Jerome Lederer, Willie Hammer, and many others to cope with the increased level of complexity in aerospace systems, particularly in military aircraft and intercontinental ballistic missile systems.

Because my last book took seven years to write and I wanted to make changes and updates starting soon after publication, I have decided to take a different approach. Instead of waiting five years for this book to appear, I am going to take advantage of new technology to create a "living book." The first chapters will be available for download from the web as soon as they are completed, and I will continue to update them as we learn more. Chapters on new techniques and approaches to engineering for safety based on the new accident model will be added as we formulate and evaluate them on real systems. Those who request it will be notified when updates are made. In order to make this approach to publication feasible, I will retain the copyright instead of assigning it to a publisher. For those who prefer or need a bound version, they will be available.

My first book, *Safeware*, forms the basis for understanding much of what is contained in this present book and provides a broader overview of the general topics in system safety engineering. The reader who is new to system safety or has limited experience in practicing it on complex systems is encouraged to read *Safeware* before they try to understand the approaches in this book. To avoid redundancy, basic information in *Safeware* will in general not be repeated, and thus *Safeware* acts as a reference for the material here. To make this book coherent in itself, however, there is some repetition, particularly on topics for which my understanding has advanced since writing *Safeware*.

Currently this book contains:

- Background material on traditional accident models, their limitations, and the goals for a new model;
- The fundamental ideas in system theory upon which the new model (as well as system engineering and system safety in particular) are based;
- A description of the model;
- An evaluation and demonstration of the model through its application to the analysis of some recent complex system accidents.

Future chapters are planned to describe novel approaches to hazard analysis, accident prevention, risk assessment, and performance monitoring, but these ideas are not yet developed adequately to justify wide dissemination. Large and realistic examples are included so the reader can see how this approach to system safety can be applied to real systems. Example specifications and analyses that are too big to include will be available for viewing and download from the web.

Because the majority of users of *Safeware* have been engineers working on real projects, class exercises and other teaching aids are not included. I will develop such teaching aids for my own classes, however, and they, as well as a future self-instruction guide for learning this approach to system safety, will be available as they are completed. Course materials will be available under the new MIT Open Courseware program to make such materials freely available on the web (http://ocw.mit.edu).

Contents

[1] Parts of this section were written by Mirna Daouk, Nicolas Dulac, and Karen Marais, who created the systems dynamics model of the Walkerton accident. We are working on making these models easier to read and to create so they can be used more widely by engineers without extensive training but the results of this research are not yet ready for this book and will be added later.

[2] A version of this appendix is in press for publication in the AIAA Journal of Spacecraft and Rockets.

Part I

Foundations

Chapter 1

Why Do We Need a New Model?

Accident models underlie all efforts to engineer for safety: they form the basis for (1) investigating and analyzing accidents; (2) designing to prevent future losses; and (3) determining whether systems are suitable for use by assessing the risk associated with an activity, the use of a product, or the operation of a system. While you may not be consciously aware you are using a model when engaged in these activities, some (perhaps subconscious) model of the phenomenon is always part of the process.

Models are important because they provide a means for understanding phenomena and recording that understanding in a way that can be communicated to others. Note that all models are abstractions—they simplify the thing being modeled by abstracting away what are assumed to be irrelevant details and focusing on the features of the phenomenon that are assumed to be the most relevant. That selection process in most cases is arbitrary and dependent entirely on the choice of the modeler, but it is critical in determining the usefulness and accuracy of the model in predicting future events.

Accident models are used to explain how accidents occur. An underlying assumption of all accident models is that there are common patterns in accidents and that they are not simply random events. Accident models impose patterns on accidents and influence the factors considered in any safety analysis. Therefore, the model used may either act as a filter and bias toward considering only certain events and conditions or it may expand activities by forcing consideration of factors that are often omitted. Because the accident model influences what cause(s) is ascribed to an accident, the countermeasures taken to prevent future accidents, and the evaluation of the risk in operating a system, the power and features of the model will greatly affect our ability to identify and control hazards and thus prevent accidents.

The most common system safety accident models have their roots in industrial safety and view accidents as resulting from a chain or sequence of events. Such models work well for losses caused by failures of physical components and for relatively simple systems. But since World War II, the types of systems we are attempting to build and the context in which they are being built has been changing. These changes are stretching the limits of current accident models and safety engineering techniques:

- **Fast pace of technological change**: Technology is changing faster than the engineering techniques to cope with the new technology are being created. Lessons learned over centuries about designing to prevent accidents may be lost or become ineffective when older technologies are replaced with new ones. New technology introduces unknowns into our systems and even *unk-unks* (unknown unknowns).

At the same time that the development of new technology has sprinted forward, the time to market for new products has significantly decreased and strong pressures exist to decrease this time even further. The average time to translate a basic technical discovery into a commercial product in the early part of this century was 30 years. Today our technologies get to market in two to three years and may be obsolete in five. We no longer have the luxury of carefully testing systems and designs to understand all the potential behaviors and risks before commercial or scientific use.

- **Changing Nature of Accidents**: Digital technology has created a quiet revolution in most fields of engineering, but system engineering and system safety engineering techniques have not kept pace. Digital systems are changing the nature of accidents. Many of the approaches to prevent accidents that worked on electromechanical components—such as replication of components to protect against individual component failure—are ineffective in controlling accidents that arise from the use of digital systems and software. Overconfidence in redundancy and misunderstanding of the "failure" modes of software-implemented functions has played an important role in recent aerospace accidents [67], such as the loss of the Ariane 5 on its first flight.

- **New types of hazards**: The most common accident models are based on an underlying assumption that accidents are the result of an uncontrolled and undesired release of energy or interference in the normal flow of energy. Our increasing dependence on information systems are, however, creating the potential for loss of information or incorrect information that can lead to unacceptable physical, scientific, or financial losses. The "head in the sand" approach of simply denying that software is safety-critical when it only provides information and does not directly release energy (a common attitude in some applications, such as air traffic control) is becoming less and less acceptable as software plays an increasingly important role in accidents.

- **Increasing complexity and coupling**: Complexity has many facets, most of which are increasing in the systems we are building, particularly *interactive* complexity. We are designing systems with potential interactions among the components that cannot be thoroughly planned, understood, anticipated, or guarded against. The operation of some systems is so complex that it defies the understanding of all but a few experts, and sometimes even they have incomplete information about its potential behavior. Software is an important factor here—it has allowed us to implement more integrated, multi-loop control in systems containing large numbers of dynamically interacting components where tight coupling allows disruptions or dysfunctional interactions in one part of the system to have far-ranging rippling effects. The problem is that we are attempting to build systems that are beyond our ability to intellectually manage: Increased interactive complexity and coupling make it difficult for the designers to consider all the potential system states or for operators to handle all normal and abnormal situations and disturbances safely and effectively. This situation is not new. Throughout history, inventions and new technology have often gotten ahead of their scientific underpinnings and engineering knowledge, but the result has always been increased risk and accidents until science and engineering caught up.[1]

- **Decreasing tolerance for single accidents**: The losses stemming from accidents is increasing with the cost and potential destructiveness of the systems we build. Our new scientific and technological discoveries have not only created new or increased hazards (such as radiation exposure and chemical pollution) but have provided the means to harm increasing numbers of people as the scale of our systems increases and to impact future generations through environmental pollution and genetic damage. Financial losses and lost potential for scientific advances are also increasing in an age where, for example, a spacecraft may take 10 years and up to a billion dollars to build. Learning from accidents (the *fly-fix-fly* approach to safety) needs to be supplemented with increasing emphasis on preventing the first one.

- **More complex relationships between humans and automation**: Humans are increasingly sharing control of systems with automation and moving into positions of higher-level decision making with automation implementing the decisions. These changes are leading to new types of human error—such as new types of mode confusion—and a new distribution of human errors, for example, increasing errors of omission versus commission [105, 106]. All human behavior is influenced by the context in which it occurs, and operators in high-tech systems are often at the mercy of the design of the automation they use. Many recent accidents that have been blamed on operator error could more accurately be labeled as resulting from flawed system, software, and interface design. Inadequacies in communication between humans and machines is becoming an increasingly important factor in accidents.

- **Changing regulatory and public views of safety**: In today's complex and interrelated societal structure, responsibility for safety is shifting from the individual to government. Individuals no longer have the ability to control the risks around them and are demanding that government assume greater responsibility for controlling behavior through laws and various forms of oversight and regulation. As companies come under increasing pressure to satisfy time-to-market and budgetary pressures, government will have to step in to provide the protection the public demands. The alternative is individuals and groups turning to the courts for protection, which could have much worse potential effects, such as unnecessarily stifling innovation through fear of law suits.

These changes are challenging both our accident models and the accident prevention and risk assessment techniques based on them. Hazard analysis and engineering techniques are needed that deal with the new types of systems we are building. This book suggests that systems theory is an appropriate foundation for such techniques, presents a new accident model based upon it, and describes new accident analysis and prevention procedures based on the new model.

[1] As an example, consider the introduction of high-pressure steam engines in the first half of the nineteenth century, which transformed industry and transportation but resulted in frequent and disastrous explosions. While engineers quickly amassed scientific information about thermodynamics, the action of steam in the cylinder, the strength of materials in the engine and many other aspects of steam engine operation, there was little scientific understanding about the buildup of steam pressure in the boiler, the effect of corrosion and decay, and the causes of boiler explosions. High-pressure steam had made the current boiler design obsolete by producing excessive strain on the boilers and exposing weaknesses in the materials and construction. Attempts to add technological safety devices were unsuccessful because engineers did not fully understand what went on in steam boilers: It was not until well after the mid-century that the dynamics of steam generation was understood. For an examination of the parallels between the early development of high-pressure steam engines and software engineering, see [65].

Part I provides the foundation for this model. Chapter 2 reviews the limitations of current event-based accident models while Chapter 3 outlines the goals and capabilities necessary to handle the new types of systems we are building and the new technology with which we are building them. Chapter 4 presents some basic concepts in systems theory and describes how safety fits into system engineering.

Chapter 2

Limitations of Traditional Accident Models

2.1 Understanding Accidents using Event Chains

The earliest formal accident models came from *industrial safety* (sometimes called *operational safety*) and reflect the factors inherent in protecting workers against industrial accidents. Later, these same models and variants of them were applied to the engineering and operation of complex systems, called *system safety.* At the beginning, the focus in industrial accident prevention was on unsafe conditions, such as open blades and unprotected belts. While this emphasis on preventing unsafe conditions was very successful in reducing industrial injuries, the decrease naturally started to slow down as the most obvious accident factors were eliminated, and the emphasis shifted to unsafe acts: Accidents began to be regarded as someone's fault rather than as an event that could have been prevented by some change in the plant or product.

Heinrich's Domino Model, published in 1931, was one of the first general accident models and was very influential in shifting the emphasis to human error. Heinrich compared the general sequence of accidents to five dominoes, standing on end in a line (Figure 2.1). When the first domino falls, it automatically knocks down its neighbor and so on until the injury occurs. In any accident sequence, according to this model, ancestry or social environment leads to a fault of a person, which is the proximate reason for an unsafe act or condition (mechanical or physical), which results in an accident, which leads to an injury. The Domino Model proved to be inadequate for complex systems and other models were developed, but the assumption that there is a cause of an accident unfortunately persists.

The most common accident models today explain accidents in terms of multiple events sequenced as a forward chain over time. The events considered as critical in these models almost always involve some type of component failure or human error, or are energy related. The chains may be branching (as in fault trees) or there may be multiple chains synchronized by time or common events [9]. Lots of notations have been developed to represent the events in a graphical form, but the underlying model is the same. Figure 2.2 shows an example for the rupture of a pressurized tank. Formal and informal notations for representing the event chain may contain the events only or they may also contain the conditions that led to the events. Events create conditions that, along with existing conditions, lead to events that create new conditions and so on.

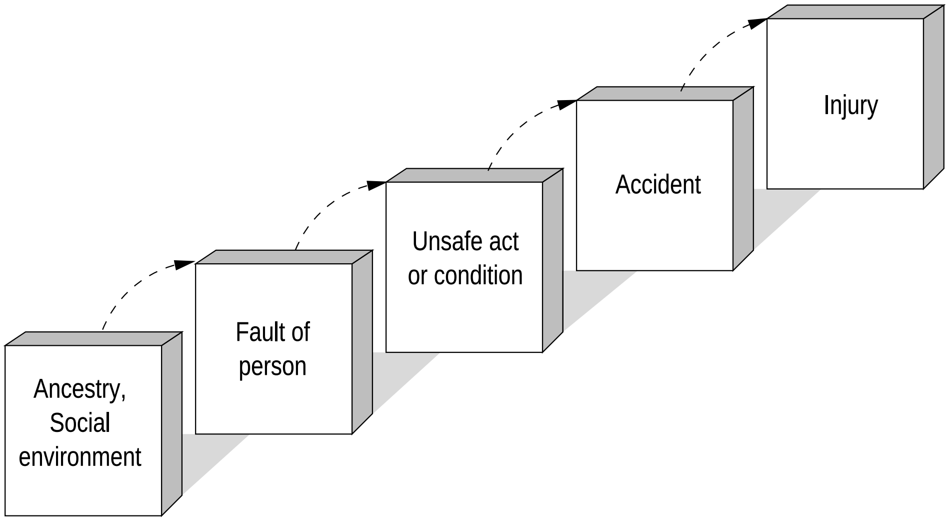


Figure 2.1: Heinrich's Domino Model of Accidents.

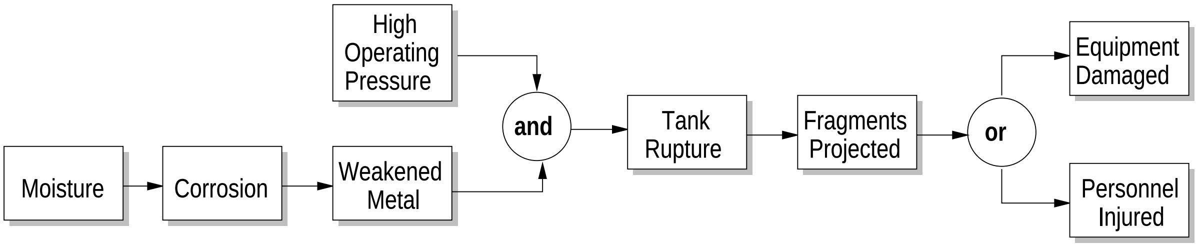


Figure 2.2: A model of the chain of events leading to the rupture of a pressurized tank (adapted from Hammer [42]). Moisture leads to corrosion, which causes weakened metal, which together with high operating pressures causes the tank to rupture, resulting in fragments being projected, and finally leading to personnel injury and/or equipment failure.

The causal relationships between the events in the chain are direct and linear, representing the notion that the preceding event or condition must have been present for the subsequent event to occur, i.e., if event X had not occurred then the following event Y would not have occurred. As such, event chain models encourage limited notions of linear causality, and it is difficult or impossible to incorporate non-linear relationships, including feedback. Other examples of event-chain models are described in *Safeware* (Chapter 10).

In event-based models, the causal factors identified depend on the events that are considered and on the selection of the conditions related to those events, i.e., the conditions that link the events. However, other than the physical events immediately preceding or directly involved in the loss, the choice of events to include is subjective and the conditions selected to explain the events is even more so. Event chain models also limit the countermeasures chosen to prevent losses and the quantification of risk associated with the operation of the system.

2.1.1 Selecting Events

The selection of events to include in an event chain is dependent on the stopping rule used to determine how far back the sequence of explanatory events goes. Although the first event in the chain is often labeled the *initiating event* or *root cause*, the selection of an initiating event is arbitrary and previous events and conditions could always be added.

Sometimes the initiating event is selected (the backward chaining stops) because it represents a type of event that is familiar and thus acceptable as an explanation for the accident. In other cases, the initiating event or root cause is chosen because it is the first event in the backward chain for which it is felt that something can be done for correction.[1] The backward chaining may also stop because the causal path disappears due to lack of information. Rasmussen suggests that a practical explanation for why actions by operators actively involved in the dynamic flow of events are so often identified as the cause of an accident is the difficulty in continuing the backtracking "through" a human. Operator actions are often selected as the stopping point in an accident event chain. Finally, events may be excluded or not examined in depth because they raise issues that are embarrassing to the organization or its contractors or are politically unacceptable.

As just one example, the accident report on a friendly fire shootdown of a helicopter over the Iraqi No-Fly-Zone in 1994 describes the accident as a chain of events leading to the shootdown. Included in the chain of events provided is the fact that the helicopter pilots did not change to the radio frequency required in the No-Fly-Zone when they entered it (they stayed on the enroute frequency). Stopping at this event in the chain, it appears that the helicopter pilots were partially at fault for the loss by making an important mistake. An independent account of the accident [90], however, notes that the U.S. Commander of the operation had made an exception about the radio frequency to be used by the helicopters in order to mitigate a different safety concern (see Chapter 7.3), and therefore the pilots were simply following orders. This commanded exception to radio procedures is not included in the chain of events included in the official government accident report, but it provides a very different understanding of the role of the helicopter pilots in the loss.

The problem with event chain models is not simply that the selection of the events to include is arbitrary. Even more important is that viewing accidents as chains of events may limit understand-

[1] As an example, a NASA Procedures and Guidelines document (NPG 8621 Draft 1) defines a root cause as: "Along a chain of events leading to an mishap, the first causal action or failure to act that could have been controlled systematically either by policy/practice/procedure or individual adherence to policy/practice/procedure."

ing and learning from the loss. Event chains developed to explain an accident usually concentrate on the proximate events immediately preceding the loss. But the foundation for an accident is often laid years before. One event simply triggers the loss, but if that event had not happened, another one would have. The Bhopal disaster provides a good example.

The release of methyl isocyanate (MIC) from the Union Carbide chemical plant in Bhopal, India, in December 1984 has been called the worst industrial accident in history: Conservative estimates point to 2000 fatalities, 10,000 permanent disabilities (including blindness), and 200,000 injuries [19]. The Indian government blamed the accident on human error—the improper cleaning of a pipe at the plant. A relatively new worker was assigned to wash out some pipes and filters, which were clogged. MIC produces large amounts of heat when in contact with water, and the worker properly closed the valves to isolate the MIC tanks from the pipes and filters being washed. Nobody, however, inserted a required safety disk (called a slip blind) to back up the valves in case they leaked [7].

A chain of events describing the accident mechanism for Bhopal might include:

E1 Worker washes pipes without inserting a slip blind.[2]

E2 Water leaks into MIC tank.

E3 Explosion occurs.

E4 Relief valve opens.

E5 MIC vented into air.

E6 Wind carries MIC into populated area around plant.

A different operator error might be identified as the root cause (initiating event) if the chain is followed back farther. The worker who had been assigned the task of washing the pipes reportedly knew that the valves leaked, but he did not check to see whether the pipe was properly isolated because, he said, it was not his job to do so. Inserting the safety disks was the job of the maintenance department, but the maintenance sheet contained no instruction to insert this disk. The pipe-washing operation should have been supervised by the second shift supervisor, but that position had been eliminated in a cost-cutting effort.

But the selection of a stopping point and the specific operator error to label as the root cause is not the real problem here—it is the limitations implicit in using a chain of events to understand why this accident occurred: Given the design and operating conditions of the plant, an accident was waiting to happen:

> However [water] got in, it would not have caused the severe explosion had the refrigeration unit not been disconnected and drained of freon, or had the gauges been properly working and monitored, or had various steps been taken at the first smell of MIC instead of being put off until after the tea break, or had the scrubber been in service, or had the water sprays been designed to go high enough to douse the emissions, or had

[2] Union Carbide lawyers have argued that the introduction of water into the MIC tank was an act of sabotage rather than a maintenance worker mistake. While this differing interpretation of the initiating event has important implications with respect to legal liability, it makes no difference in the argument presented here regarding the limitations of event-chain models of accidents or even, as will be seen, understanding why this accident occurred.

> the flare tower been working and been of sufficient capacity to handle a large excursion [88, p.349].

It is not uncommon for a company to turn off passive safety devices, such as refrigeration units, to save money. In addition, gauges at plants are frequently out of service [12]. At the Bhopal facility, there were few alarms or interlock devices in critical locations that might have warned operators of abnormal conditions—a system design deficiency.

The operating manual specified that the refrigeration unit *must* be operating whenever MIC was in the system: The chemical has to be maintained at a temperature no higher than 5° Celsius to avoid uncontrolled reactions. A high temperature alarm was to sound if the MIC reached 11°. The refrigeration unit was turned off, however, to save money and the MIC was usually stored at nearly 20°. The plant management adjusted the threshold of the alarm, accordingly, from 11° to 20° and logging of tank temperatures was halted, thus eliminating the possibility of an early warning of rising temperatures.

Other protection devices at the plant had inadequate design thresholds. The vent scrubber, had it worked, was designed to neutralize only small quantities of gas at fairly low pressures and temperatures: The pressure of the escaping gas during the accident exceeded the scrubber's design by nearly two and a half times, and the temperature of the escaping gas was at least 80° Celsius more than the scrubber could handle. Similarly, the flare tower (which was supposed to burn off released vapor) was totally inadequate to deal with the estimated 40 tons of MIC that escaped during the accident. In addition, the MIC was vented from the vent stack 108 feet above the ground, well above the height of the water curtain intended to knock down the gas. The water curtain reached only 40 to 50 feet above the ground. The water jets could reach as high as 115 feet, but only if operated individually.

Leaks were routine occurrences and the reasons for them were seldom investigated: Problems were either fixed without further examination or were ignored. A safety audit two years earlier by a team from Union Carbide had noted many safety problems at the plant, including several involved in the accident, such as filter-cleaning operations without using slip blinds, leaking valves, the possibility of contaminating the tank with material from the vent gas scrubber, and bad pressure gauges. The safety auditors had recommended increasing the capability of the water curtain and had pointed out that the alarm at the flare tower from which the MIC leaked was nonoperational and thus any leak could go unnoticed for a long time. None of the recommended changes were made [12]. There is debate about whether the audit information was fully shared with the Union Carbide India subsidiary and about who was responsible for making sure changes were made. In any event, there was no follow-up to make sure that the problems had been corrected. A year before the accident, the chemical engineer managing the MIC plant resigned because he disapproved of falling safety standards and still no changes were made. He was replaced by an electrical engineer.

Measures for dealing with a chemical release once it occurred were no better. Alarms at the plant sounded so often (the siren went off 20 to 30 times a week for various purposes) that an actual alert could not be distinguished from routine events or practice alerts. Ironically, the warning siren was not turned on until two hours after the MIC leak was detected (and after almost all the injuries had occurred) and then was turned off after only five minutes—which was company policy [7]. Moreover, the numerous practice alerts did not seem to be effective in preparing for an emergency: When the danger during the release became known, many employees ran from the contaminated areas of the plant, totally ignoring the buses that were sitting idle ready to evacuate workers and nearby residents. Plant workers had only a bare minimum of emergency equipment—a shortage of oxygen masks, for example, was discovered after the accident started—and they had almost no knowledge or training about how to handle nonroutine events.

The police were not notified when the chemical release began; in fact, when called by police and reporters, plant spokesmen first denied the accident and then claimed that MIC was not dangerous. Nor was the surrounding community warned of the dangers, before or during the release, or informed of the simple precautions that could have saved them from lethal exposure, such as putting a wet cloth over their face and closing their eyes. If the community had been alerted and provided with this simple information, many (if not most) lives would have been saved and injuries prevented [60].

These are only a few of the factors involved in this catastrophe, which include other technical and human errors within the plant, design errors, management negligence, regulatory deficiencies on the part of the U.S. and Indian governments, and general agricultural and technology transfer policies related to the reason they were making such a dangerous chemical in India in the first place. Any one of these perspectives or "causes" is inadequate by itself to understand the accident and to prevent future ones. In particular, identifying only operator error (failing to insert the slip blind) as the root cause of the accident (the first or initiating event in the event chain) ignores most of the opportunities for the prevention of this and similar accidents in the future.

The maintenance worker was, in fact, only a minor and somewhat irrelevant player in the loss. Instead, degradation in the safety margin occurred over time and without any particular single decision to do so but simply as a series of decisions that moved the plant slowly toward a situation where any slight error would lead to a major accident. Given the overall state of the Bhopal Union Carbide plant and its operation, if the action of inserting the slip disk had not been left out of the pipe washing operation that December day in 1984, something else would have triggered an accident. In fact, a similar leak had occurred the year before, but did not have the same catastrophic consequences and the true root causes of that incident were not identified nor fixed.

To identify one event (such as a maintenance worker leaving out the slip disk) or even several events as the root cause or the start of an event chain leading to the Bhopal accident is misleading at best. Rasmussen writes:

> The stage for an accidental course of events very likely is prepared through time by the normal efforts of many actors in their respective daily work context, responding to the standing request to be more productive and less costly. Ultimately, a quite normal variation in somebody's behavior can then release an accident. Had this 'root cause' been avoided by some additional safety measure, the accident would very likely be released by another cause at another point in time. In other words, an explanation of the accident in terms of events, acts, and errors is not very useful for design of improved systems [96].

2.1.2 Selecting Conditions

In addition to subjectivity in selecting the events, the links between the events that are chosen to explain them are subjective and subject to bias. Leplat notes that the links are justified by knowledge or rules of different types, including physical and organizational knowledge. The same event can give rise to different types of links according to the mental representations the analyst has of the production of this event. When several types of rules are possible, the analyst will remember those that agree with his or her mental model of the situation [62].

In the loss of an American Airlines B-757 near Cali, Colombia, in 1995, two significant events were (1) *Pilot asks for clearance to take the Rozo approach* followed later by (2) *Pilot types R into the FMS.*[3] In fact, the pilot should have typed the four letters *ROZO* instead of *R*—the latter was the symbol for a different radio beacon (called Romeo) near Bogota—and as a result the aircraft incorrectly turned toward mountainous terrain. While these events are noncontroversial, the link between the two events could be explained by any of the following:

- *Pilot Error*: In the rush to start the descent, the pilot executed a change of course without verifying its effect on the flight path.
- *Crew Procedure Error*: In the rush to start the descent, the captain entered the name of the waypoint without normal verification from the other pilot.
- *Approach Chart and FMS Inconsistencies*: The identifier used to identify Rozo on the approach chart (*R*) did not match the identifier used to call up Rozo in the FMS.
- *FMS Design Deficiency*: The FMS did not provide the pilot with feedback that choosing the first identifier listed on the display was not the closest beacon with that identifier.
- *American Airlines Training Deficiency*: The pilots flying into South America were not warned about duplicate beacon identifiers nor adequately trained on the logic and priorities used in the FMS on the aircraft.
- *Manufacturer Deficiency*: Jeppesen-Sanderson did not inform airlines operating FMS-equipped aircraft of the differences between navigation information provided by Jeppesen-Sanderson Flight Management System navigation databases and Jeppesen-Sanderson approach charts or the logic and priorities employed in the display of electronic FMS navigation information.
- *International Standards Deficiency*: No single worldwide standard provides unified criteria for the providers of electronic navigation databases used in Flight Management Systems.

The selection of the linking condition will greatly influence the cause ascribed to the accident yet in the example all are plausible and each could serve as an explanation of the event sequence. The choice may reflect more on the person or group making the selection than on the accident itself. In fact, understanding this accident and learning enough from it to prevent future accidents requires identifying all these factors: The accident model used should encourage and guide a comprehensive analysis at multiple technical and social system levels.

2.1.3 Selecting Countermeasures

The use of event-chain models has important implications for the way engineers design for safety. If an accident is caused by a chain of events, then the most obvious preventive measure is to break the chain so the loss never occurs. Breaking the chain requires either removing the events or conditions (preventing them from occurring) or adding enough AND gates (required simultaneous conditions or events) that the likelihood of the chaining factors being realized is very low, that is, the accident sequence is broken. Because the most common events considered in event-based

[3] An FMS is an automated Flight Management System, which assists the pilots in various ways. In this case, it was being used to provide navigation information.

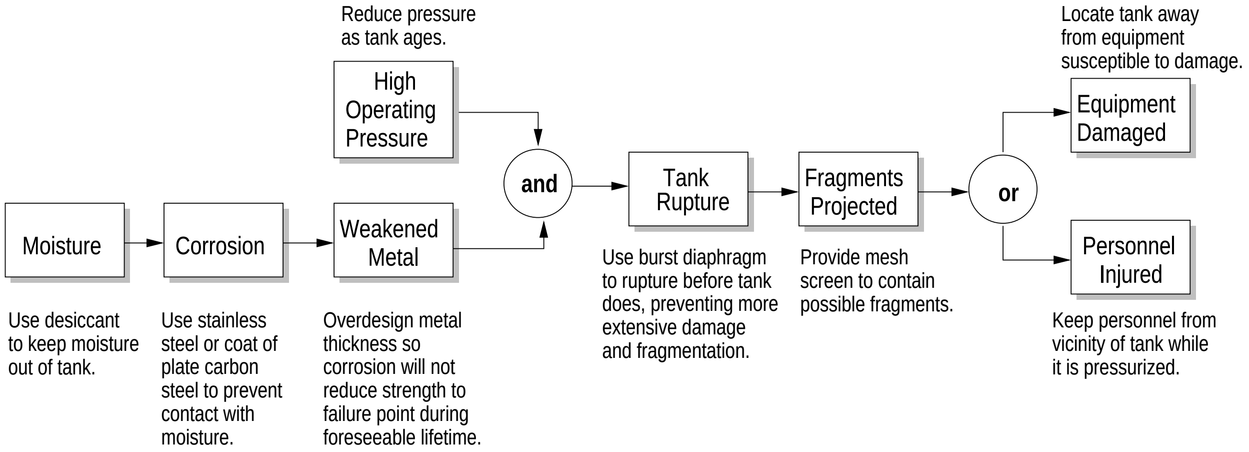

Reduce pressure as tank ages.
High Operating Pressure
Locate tank away from equipment susceptible to damage.
Equipment Damaged
and
Tank Rupture
Fragments Projected
or
Moisture
Corrosion
Weakened Metal
Personnel Injured
Use burst diaphragm to rupture before tank does, preventing more extensive damage and fragmentation.
Provide mesh screen to contain possible fragments.
Use desiccant to keep moisture out of tank.
Use stainless steel or coat of plate carbon steel to prevent contact with moisture.
Overdesign metal thickness so corrosion will not reduce strength to failure point during foreseeable lifetime.
Keep personnel from vicinity of tank while it is pressurized.

Figure 2.3: The pressurized tank rupture event chain along with measures that could be taken to “break” the chain by preventing individual events in it.

models are component failures, preventive measures tend to be focused on preventing failure events (i.e., increasing component integrity) and on reducing their likelihood of leading to an accident by adding redundancy (AND gates) to the design.

As an example, Figure 2.3 contains the event chain for the rupture of a pressurized tank, shown earlier, augmented with mitigation measures designed to break the chain. For this simple example, involving only physical failures, the approach works well. But even this simple example omits any consideration of factors only indirectly related to the events in the chain, for example, competitive or financial pressures to increase efficiency that could lead to not following plans to reduce the operating pressure as the tank ages.

The common focus on component failures as the cause of accidents and the use of event chain models has led to an overreliance on redundancy as protection against losses. Redundancy does not provide protection against the types of accidents arising with the use of new technology, particularly digital technology [58]. These accidents often arise from dysfunctional interactions among components, where no component may have actually failed. In fact, redundancy may exacerbate the problems and actually contribute to system complexity and accidents. A NASA study of an experimental aircraft executing two versions of the control system found that all of the software problems occurring during flight testing resulted from errors in the redundancy management system and not in the much simpler control software itself, which worked perfectly [75]. Redundancy is not particularly effective in protecting against human errors either: pressures to work more efficiently usually cause such redundancy to deteriorate over time.

With the increasing role of software and humans in supervisory control of complex systems, concentrating our safety engineering efforts on physical failures and the use of redundancy to prevent them will become increasingly ineffective. The Ariane 5 loss is an apt reminder of this fact: both the primary and backup computers shut themselves down when a data exception occurred.

2.1.4 Assessing Risk

Limitations of event-chain models are also reflected in the current approaches to quantitative risk assessment. When the goal of the analysis is to perform a probabilistic risk assessment (PRA), initiating events in the chain are usually assumed to be mutually exclusive. While this assumption simplifies the mathematics, it may not match reality.

As an example, consider the following description of a accident chain for an offshore oil platform:

> An initiating event is an event that triggers an accident sequence—e.g., a wave that exceeds the jacket's capacity that, in turn, triggers a blowout that causes failures of the foundation. As initiating events, they are mutually exclusive; only one of them starts the accident sequence. A catastrophic platform failure can start by failure of the foundation, failure of the jacket, or failure of the deck. These initiating failures are also (by definition) mutually exclusive and constitute the basic events of the [Probabilistic Risk Assessment] model in its simplest form [85, p.121].

The selection of the failure of the foundation, jacket, or deck as the initiating event is arbitrary, as we have seen, and eliminates from consideration prior events leading to them such as manufacturing or construction problems. The failure of the foundation, for example, might be related to the use of inferior construction materials which, in turn, may be related to budget deficiencies or lack of government oversight.

In addition, there does not seem to be any reason for assuming that initiating failures are mutually exclusive and that only one starts the accident, except perhaps to simplify the mathematics. In accidents, seemingly independent failures may have a common systemic cause (often not a failure) that results in coincident failures. For example, the same pressures to use inferior materials in the foundation may result in their use in the jacket and the deck, leading to a wave causing coincident, dependent failures in all three. Alternatively, the design of the foundation (a systemic factor rather than a failure event) may lead to pressures on the jacket and deck when stresses cause deformities in the foundation. Treating such events as independent may lead to unrealistic risk assessments.

In the Bhopal accident, the vent scrubber, flare tower, water spouts, refrigeration unit, and various monitoring instruments were all out of operation simultaneously. Assigning probabilities to all these seemingly unrelated events and assuming independence would lead one to believe that this accident was merely a matter of a once-in-a-lifetime coincidence. A probabilistic risk assessment based on an event chain model most likely would have treated these conditions as independent failures and then calculated their coincidence as being so remote as to be beyond consideration. On the surface, it does seem incredible that these devices were all out of operation simultaneously. However, a closer look paints a quite different picture and shows these were not random failure events but were related to common engineering design and management decisions.

Most accidents in well-designed systems involve two or more low-probability events occurring in the worst possible combination. When people attempt to predict system risk, they explicitly or implicitly multiply events with low probability—assuming independence—and come out with impossibly small numbers, when, in fact, the events are dependent. This dependence may be related to common systemic factors that do not appear in an event chain. Machol calls this phenomenon the *Titanic coincidence* [74].[4]. A number of "coincidences" contributed to the Titanic accident and the subsequent loss of life. For example, the captain was going far too fast for existing conditions, a proper watch for icebergs was not kept, the ship was not carrying enough lifeboats, lifeboat drills were not held, the lifeboats were lowered properly but arrangements for manning

[4] Watt defined a related phenomenon he called the *Titanic effect* to explain the fact that major accidents are often preceded by a belief that they cannot happen. The Titanic effect says that the magnitude of disasters decreases to the extent that people believe that disasters are possible and plan to prevent them or to minimize their effects [119]

them were insufficient, and the radio operator on a nearby ship was asleep and so did not hear the distress call. Many of these events or conditions may be considered independent but appear less so when we consider that overconfidence due to incorrect engineering analyses about the safety and unsinkability of the ship most likely contributed to the excessive speed, the lack of a proper watch, and the insufficient number of lifeboats and drills. That the collision occurred at night contributed to the iceberg not being easily seen, made abandoning ship more difficult than it would have been during the day, and was a factor in why the nearby ship's operator was asleep [77]. Assuming independence here leads to a large underestimate of the true risk.

Another problem in probabilistic risk assessment (PRA) is the emphasis on failure events—design errors are usually omitted and only come into the calculation indirectly through the probability of the failure event. Accidents involving dysfunctional interactions among non-failing (operational) components are usually not considered. Systemic factors also are not reflected. In the offshore oil platform example above, the true probability density function for the failure of the deck might reflect a poor design for the conditions the deck must withstand (a human design error) or, as noted above, the use of inadequate construction materials due to lack of government oversight or project budget limitations.

When historical data is used to determine the failure probabilities used in the PRA, non-failure factors, such as design errors or unsafe management decisions may differ between the historic systems from which the data was derived and the system under consideration. It is possible (and obviously desirable) for each PRA to include a description of the conditions under which the probabilities were derived. If such a description is not included, it may not be possible to determine whether conditions in the platform being evaluated differ from those built previously that might significantly alter the risk. The introduction of a new design feature or of active control by a computer might greatly affect the probability of failure and the usefulness of data from previous experience then becomes highly questionable.

The most dangerous result of using PRA arises from considering only immediate physical failures. Latent design errors may be ignored and go uncorrected due to overconfidence in the risk assessment. An example, which is a common but dangerous practice judging from its implication in a surprising number of accidents, is wiring a valve to detect only that power has been applied to open or close it and not that the valve has actually operated. In one case, an Air Force system included a relief valve to be opened by the operator to protect against overpressurization [3]. A secondary valve was installed as backup in case the primary relief valve failed. The operator needed to know when the first valve had not opened in order to determine that the secondary valve must be activated. One day, the operator issued a command to open the primary valve. The position indicator and open indicator lights both illuminated but the primary relief valve was *not* open. The operator, thinking the primary valve had opened, did not activate the backup valve and an explosion occurred.

A post-accident investigation discovered that the indicator light circuit was wired to indicate *presence of power* at the valve, but it did not indicate valve *position.* Thus, the indicator showed only that the activation button had been pushed, not that the valve had operated. An extensive probabilistic risk assessment of this design had correctly assumed a low probability of simultaneous failure for the two relief valves, but had ignored the possibility of a design error in the electrical wiring: The probability of that design error was not quantifiable. If it had been identified, of course, the proper solution would have been to eliminate the design error, not to assign a probability to it. The same type of design flaw was a factor in the Three Mile Island accident: An indicator misleadingly showed that a discharge valve had been ordered closed but not that it had actually closed. In fact, the valve was blocked in an open position.

In addition to these limitations of PRA for electromechanical systems, current methods for quantifying risk that are based on combining probabilities of individual component failures and mutually exclusive events are not appropriate for systems controlled by software and by humans making cognitively complex decisions, and there is no effective way to incorporate management and organizational factors, such as flaws in the safety culture. As a result, these critical factors in accidents are often simply omitted from risk assessment because analysts do not know how to obtain a "failure" probability, or alternatively, a number is pulled out of the air for convenience. An accident model not based on failure events, such as the one introduced in this book, could provide an entirely new basis for evaluating and assessing safety.

2.2 Interpreting Events and Conditions as Causes

Although it is common to isolate one or more events or conditions and call them *the* cause (or the *proximate, direct*, or *root* cause) of an accident or incident and to label the other events or conditions as *contributory*, there is no basis for this distinction. Usually a root cause selected from the chain of events has one or more of the following characteristics: (a) it is a deviation from a standard, (b) it is accepted as a familiar and therefore reasonable explanation, (c) a "cure" is known [11], and, often, (d) it is politically acceptable as the identified cause. Making such distinctions between causes or limiting the factors considered can be a hindrance in learning from and preventing future accidents.

In the crash of an American Airlines DC-10 at Chicago's O'Hare Airport in 1979, the U.S. National Transportation Safety Board (NTSB) blamed only a "maintenance-induced crack," and not also a design error that allowed the slats to retract if the wing was punctured. Because of this omission, McDonnell Douglas was not required to change the design, leading to future accidents related to the same design error [87].

In another DC-10 saga, explosive decompression played a critical role in a near-miss over Windsor, Ontario. An American Airlines DC-10 lost part of its passenger floor, and thus all of the control cables that ran through it, when a cargo door opened in flight in June 1972. Due to the extraordinary skill and poise of the pilot, Bryce McCormick, the plane landed safely. In a remarkable coincidence, McCormick had trained himself to fly the plane using only the engines because he had been concerned about a decompression-caused collapse of the floor. After this close call, McCormick recommended that every DC-10 pilot be informed of the consequences of explosive decompression and trained in the flying techniques that he and his crew had used to save their passengers and aircraft. FAA investigators, the National Transportation Safety Board, and engineers at a subcontractor to McDonnell Douglas that designed the fuselage of the plane, all recommended changes in the design of the aircraft. Instead, McDonnell Douglas attributed the Windsor incident totally to human error on the part of the baggage handler responsible for closing the cargo compartment door (a convenient event in the event chain) and not to any error on the part of their designers or engineers and decided all they had to do was to come up with a fix that would prevent baggage handlers from forcing the door.

One of the discoveries after the Windsor incident was that the door could be improperly closed but the external signs, such as the position of the external handle, made it appear to be closed properly. In addition, this incident proved that the cockpit warning system could fail, and the crew would then not know that they were taking off without a properly closed door:

> The aviation industry does not normally receive such manifest warnings of basic design flaws in an aircraft without cost to human life. Windsor deserved to be celebrated as an exceptional case when every life was saved through a combination of crew skill and the sheer luck that the plane was so lightly loaded. If there had been more passengers and thus more weight, damage to the control cables would undoubtedly have been more severe, and it is highly questionable if any amount of skill could have saved the plane [31].

Almost two years later, in March 1974, a fully loaded Turkish Airlines DC-10 crashed near Paris resulting in 346 deaths—one of the worst accidents in aviation history. Once again, the cargo door had opened in flight, causing the cabin floor to collapse, severing the flight control cables. Immediately after the accident, Sanford McDonnell stated the official McDonnell-Douglas position that once again placed the blame on the baggage handler and the ground crew. This time, however, the FAA finally ordered modifications to all DC-10s that eliminated the hazard. In addition, an FAA regulation issued in July 1975 required all wide-bodied jets to be able to tolerate a hole in the fuselage of 20 square feet. By labeling the root cause in the event chain as baggage handler error and attempting only to eliminate that event or link in the chain rather than the basic engineering design flaws, fixes that could have prevented the Paris crash were not made.

If there are human operators in the system, they are most likely to be blamed for an accident. This phenomenon is not new. In the nineteenth century, coupling accidents on railroads were one of the main causes of injury and death to railroad workers [42]. In the seven years between 1888 and 1894, 16,000 railroad workers were killed in coupling accidents and 170,000 were crippled. Managers claimed that such accidents were due only to worker error and negligence, and therefore nothing could be done aside from telling workers to be more careful. The government finally stepped in and required that automatic couplers be installed. As a result, fatalities dropped sharply. According to the June 1896 issue of Scientific American:

> Few battles in history show so ghastly a fatality. A large percentage of these deaths were caused by the use of imperfect equipment by the railroad companies; twenty years ago it was practically demonstrated that cars could be automatically coupled, and that it was no longer necessary for a railroad employee to imperil his life by stepping between two cars about to be connected. In response to appeals from all over, the U.S. Congress passed the Safety Appliance Act in March 1893. It has or will cost the railroads $50,000,000 to fully comply with the provisions of the law. Such progress has already been made that the death rate has dropped by 35 per cent.

The tendency to blame the operator is not simply a nineteenth century problem, but persists today. During and after World War II, the Air Force had serious problems with aircraft accidents—for example, from 1952 to 1966, 7715 aircraft were lost and 8547 people killed [42]. Most of these accidents were blamed on pilots. Some aerospace engineers in the 1950s did not believe the cause was so simple and argued that safety must be designed and built into aircraft just as are performance, stability, and structural integrity. Although a few seminars were conducted and papers written about this approach, the Air Force did not take it seriously until they began to develop intercontinental ballistic missiles: there were no pilots to blame for the frequent and devastating explosions of these liquid-propellant missiles. In having to confront factors other than pilot error, the Air Force began to treat safety as a system problem, and system safety engineering programs were developed to deal with them. Similar adjustments in attitude and practice may be forced in the future by the increasing use of unmanned autonomous aircraft and other fully automated systems.

It is still common to see statements that 70% to 80% of aircraft accidents are caused by pilot error or that 85% of work accidents are due to unsafe acts by humans rather than unsafe conditions. However, closer examination of the data shows that it may be biased and incomplete: the less that is known about an accident, the most likely it will be attributed to operator error [49]. Thorough investigation of serious accidents almost invariably finds other factors. Perrow cites a U.S. Air Force study of aviation accidents that concludes that the designation of human error, or pilot error, is a convenient classification for accidents whose real cause is uncertain, complex, or embarrassing to the organization [87].

As noted earlier, operator actions represent a convenient stopping point in an event-chain model of accidents. Other reasons for the operator error statistics include: (1) operator actions are generally reported only when they have a negative effect on safety and not when they are responsible for preventing accidents; (2) blame may be based on unrealistic expectations that operators can overcome every emergency; (3) operators may have to intervene at the limits of system behavior when the consequences of not succeeding are likely to be serious and often involve a situation the designer never anticipated and was not covered by the operator's training; and (4) hindsight often allows us to identify a better decision in retrospect, but detecting and correcting potential errors before they have been made obvious by an accident is far more difficult.[5] The report on the Clapham Junction railway accident in Britain concluded:

> There is almost no human action or decision that cannot be made to look flawed and less sensible in the misleading light of hindsight. It is essential that the critic should keep himself constantly aware of that fact. [Hidden, A. (Chairman), pg. 147]

The phenomenon is so common that it has been given a name—*hindsight bias*.

All human activity takes place within and is influenced by the environment, both physical and social, in which it takes place. It is, therefore, often very difficult to separate system design error from operator error: In highly automated systems, the operator is often at the mercy of the system design and operational procedures. One of the major mistakes made by the operators at Three Mile Island was following the procedures provided to them by the utility, and the instrumentation design did not provide the information they needed to act effectively in recovering from the hazardous state [52].

In the lawsuits following the 1995 Boeing-757 Cali accident, American Airlines was held liable for the crash based on the Colombian investigators blaming crew error entirely for the accident. The official accident investigation report cited the following four causes for the loss:

1. The flightcrew's failure to adequately plan and execute the approach to runway 19 and their inadequate use of automation.

2. Failure of the flightcrew to discontinue their approach, despite numerous cues alerting them of the inadvisability of continuing the approach.

[5] The attribution of operator error as the cause of accidents is discussed more thoroughly in *Safeware* (Chapter 5).

3. The lack of situational awareness of the flightcrew regarding vertical navigation, proximity to terrain, and the relative location of critical radio aids.

4. Failure of the flightcrew to revert to basic radio navigation at a time when the FMS-assisted navigation became confusing and demanded an excessive workload in a critical phase of the flight.

Note that the blame was placed on the pilots when the automation became confusing and demanded an excessive workload rather than on the FMS itself. To be fair, the report also identifies two "contributory factors"—but *not* causes—as:

- FMS logic that dropped all intermediate fixes from the display(s) in the event of execution of a direct routing.

- FMS-generated navigational information that used a different naming convention from that published in navigational charts.

A U.S. appeals court rejected the conclusion of the report about the four causes of the accident, which led to a lawsuit by American in a federal court in which American Airlines alleged that components of the automated aircraft system made by Honeywell Air Transport Systems and Jeppesen Sanderson helped cause the crash. American blamed the software, saying Jeppesen stored the location of the Cali airport beacon in a different file from most other beacons. Lawyers for the computer companies argued that the beacon code could have been properly accessed and that the pilots were in error. The jury concluded that the two companies produced a defective product and that Jeppesen was 17 percent responsible, Honeywell was 8 percent at fault, and American was held to be 75 percent responsible. While such distribution of responsibility may be important in determining how much each company will have to pay, it does not provide any important information with respect to accident prevention in the future. The verdict is interesting, however, because the jury rejected the oversimplified notion of causality being argued. It was also one of the first cases not settled out of court where the role of software in the loss was acknowledged.

Beyond the tendency to blame operators, other types of subjectivity in ascribing cause exist. Rarely are all the causes of an accident perceived identically by everyone involved including engineers, managers, operators, union officials, insurers, lawyers, politicians, the press, the state, and the victims and their families. Such conflicts are typical in situations that involve normative, ethical, and political considerations on which people may legitimately disagree. Some conditions may be considered unnecessarily hazardous by one group yet adequately safe and necessary by another. In addition, judgments about the cause of an accident may be affected by the threat of litigation or by conflicting interests.

Research data validates this hypothesis. Various studies have found the selection of a cause(s) depends on characteristics of the victim and of the analyst (e.g., hierarchical status, degree of involvement, and job satisfaction) as well as on the relationships between the victim and the analyst and on the severity of the accident [63].

For example, one study found that workers who were satisfied with their jobs and who were integrated into and participating in the enterprise attributed accidents mainly to personal causes. In contrast, workers who were not satisfied and who had a low degree of integration and participation more often cited nonpersonal causes that implied that the enterprise was responsible [63]. Another study found differences in the attribution of accident causes among victims, safety managers, and general managers. Other researchers have suggested that accidents are attributed to factors in which the individuals are less directly involved. A further consideration may be position in the organization: The lower the position in the hierarchy, the greater the tendency to blame accidents on factors linked to the organization; individuals who have a high position in the hierarchy tend to blame workers for accidents [63].

There even seem to be differences in causal attribution between accidents and incidents: Data on near-miss (incident) reporting suggest that causes for these events are mainly attributed to technical deviations while similar events that result in losses are more often blamed on human error [32, 53].

Causal identification may also be influenced by the data collection methods. Data usually is collected in the form of textual descriptions of the sequence of events of the accident, which, as we have seen, tend to concentrate on obvious conditions or events closely preceding the accident in time and tend to leave out less obvious or indirect events and factors. There is no simple solution to this inherent bias: On the one hand, report forms that do not specifically ask for nonproximal events often do not elicit them while, on the other hand, more directive report forms that do request particular information may limit the categories of conditions considered [54].

Other factors affecting causal filtering in accident and incident reports may be related to the design of the reporting system itself. For example, the NASA Aviation Safety Reporting System (ASRS) has a category that includes non-adherence to FARs (Federal Aviation Regulations). In a NASA study of reported helicopter incidents and accidents over a nine-year period, this category was by far the largest category cited [43]. The NASA study concluded that the predominance of FAR violations in the incident data may reflect the motivation of the ASRS reporters to obtain immunity from perceived or real violations of FARs and not necessarily the true percentages.

A final complication is that human actions always involve some interpretation of the person's goals and motives. The individuals involved may be unaware of their actual goals and motivation or may be subject to various types of pressures to reinterpret their actions. Explanations by accident analysts after the fact may be influenced by their own mental models or additional goals and pressures.

Note the difference between an explanation based on goals and one based on motives: a goal represents an end state while a motive explains *why* that end state was chosen. Consider the hypothetical case where a car is driven too fast during a snow storm and it slides into a telephone pole. An explanation based on goals for this chain of events might include the fact that the driver wanted to get home quickly. An explanation based on motives might include the fact that guests were coming for dinner and the driver had to prepare the food before they arrived.

Explanations based on goals and motives depend on assumptions that cannot be *directly* measured or observed by the accident investigator. Leplat illustrates this dilemma by describing three different motives for the event *"operator sweeps the floor"*: (1) the floor is dirty; (2) the supervisor is present, or (3) the machine is broken and the operator needs to find other work [64]. Even if the people involved survive the accident, true goals and motives may not be revealed for various reasons.

Where does all this leave us? There are two basic reasons for conducting an accident investigation: (1) to assign blame for the accident and (2) to understand why it happened so that future accidents can be prevented. When the goal is to assign blame, the backward chain of events considered often stops when someone or something appropriate to blame is found, such as the baggage handler in the DC-10 case or the maintenance worker at Bhopal. As a result, the selected initiating event may provide too superficial an explanation of why the accident occurred to prevent similar losses in the future. For example, stopping at the O-ring failure in the *Challenger* accident and fixing that particular design flaw would not have eliminated the systemic flaws that could lead to accidents in the future. For *Challenger*, examples of those systemic problems include flawed decision making and the pressures that led to it, poor problem reporting, lack of trend analysis, a "silent" or ineffective safety program, communication problems, etc. None of these are "events" (although they may be manifested in particular events) and thus do not appear in the chain of events leading to the accident. Wisely, the authors of the *Challenger* accident report used an event chain only to identify the proximate physical cause and not the reasons those events occurred, and the report writers' recommendations led to many important changes at NASA or at least attempts to make such changes.[6]

Blame is not an engineering concept; it is a legal or moral one. Usually there is no objective criterion for distinguishing one factor or several factors from other factors that contribute to an accident. While lawyers and insurers recognize that many factors contribute to a loss event, for practical reasons and particularly for establishing liability, they often oversimplify the causes of accidents and identify what they call the *proximate* (immediate or direct) cause. The goal is to determine the parties in a dispute that have the legal liability to pay damages, which may be affected by the ability to pay or by public policy considerations, such as discouraging company management or even an entire industry from acting in a particular way in the future.

When learning how to engineer safer systems is the goal rather than identifying who to punish and establishing liability, then the emphasis in accident analysis needs to shift from *cause* (in terms of events or errors), which has a limiting, blame orientation, to understanding accidents in terms of *reasons*, i.e., why the events and errors occurred. In an analysis by the author of recent aerospace accidents involving software in some way, most of the reports stopped after assigning blame—usually to the operators who interacted with the software—and never got to the root of why the accident occurred, e.g., why the operators made the errors they did and how to prevent such errors in the future (perhaps by changing the software) or why the software requirements specified unsafe behavior and why that error was introduced and why it was not detected and fixed before the software was used [67].

In general, determining the relative importance of various factors to an accident may not be useful in preventing future accidents. Haddon [41] argues, reasonably, that countermeasures to accidents should *not* be determined by the relative importance of the causal factors; instead, priority should be given to the measures that will be most effective in reducing future losses. Explanations involving events in an event chain often do not provide the information necessary to prevent future losses, and spending a lot of time determining the relative contributions of events or conditions to accidents (such as arguing about whether an event is the root cause or a contributory cause) is not productive outside the legal system. Rather, engineering effort should be devoted to identifying the factors (1) that are easiest or most feasible to change, (2) that will prevent large classes of accidents, and (3) over which we have the greatest control,

Because the goal of the model and associated system safety engineering techniques described in this book is understanding and preventing accidents rather than assigning blame, the emphasis is

[6] Recently, another Space Shuttle has been lost. While the proximate cause for the *Columbia* accident (foam hitting the wing of the orbiter) was very different than for *Challenger*, many of the systemic or root causes were similar and reflected either inadequate fixes of these factors after the *Challenger* accident or their re-emergence in the years between these losses.

on identifying the factors involved in an accident and understanding the relationship among these factors to provide an explanation of why the accident occurred. There is no attempt to determine which factors are more “important” than others but rather how they relate to each other and to the final loss event or near miss. Models based on event chains are not the most effective way to accomplish this goal.

Chapter 3

Extensions Needed to Traditional Models

Event-based models work best for accidents where one or several physical components fail, leading to a system failure or hazard. However, such models and the explanations based on them can easily miss subtle and complex couplings and interactions among failure events and omit entirely accidents involving no component failure at all. New models that are more effective for accidents in complex systems will need to account for social and organizational factors, system accidents and dysfunctional interactions, human error and flawed decision making, software errors, and adaptation.

3.1 Social and Organizational Factors

Event-based models are poor at representing systemic accident factors such as structural deficiencies in the organization, management deficiencies, and flaws in the safety culture of the company or industry. An accident model should encourage a broad view of accident mechanisms that expands the investigation beyond the proximate events: A narrow focus on technological components and pure engineering activities may lead to ignoring some of the most important factors in terms of preventing future accidents.

As an example, an explosion at a chemical plant in Flixborough, Great Britain, in June 1974, resulted in 23 deaths, 53 injuries, and $50 million in damages to property (including 2,450 houses) up to five miles from the site of the plant.[1] The official accident investigators devoted most of their effort to determining which of two pipes was the first to rupture (Figure 3.1). The British Court of Inquiry concluded that "the disaster was caused by a coincidence of a number of unlikely errors in the design and installation of a modification [a bypass pipe]," and "such a combination of errors is very unlikely ever to be repeated" [25].

While these conclusions are true if only specific technical failures and immediate physical events are considered, they are not true if a broader view of accidents is taken. The pipe rupture at Flixborough was only a small part of the cause of this accident. A full explanation and prevention

[1] Many more lives might have been lost had the explosion not occurred on a weekend when only a small shift was at the plant, the wind was light, and many of the nearby residents were away at a Saturday market in a neighboring town.

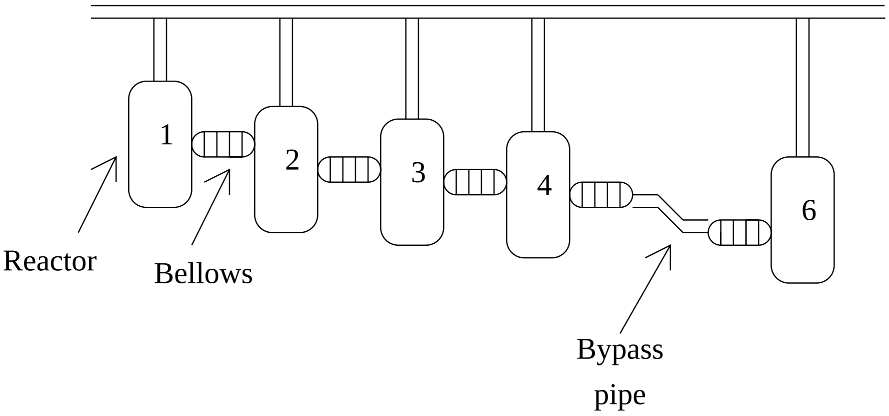


Figure 3.1: The temporary change at Flixborough. A bypass pipe was used to replace a reactor that had been removed to repair a crack. The crack had resulted from a tripling of the process capacity.

of future such accidents requires an understanding of the management practices of running the Flixborough plant without a qualified engineer on site, allowing unqualified personnel to make important modifications to the equipment, making engineering changes without properly evaluating their safety, and storing large quantities of chemicals under dangerous conditions [22]. The British Court of Inquiry investigating the accident amazingly concluded that “there were undoubtedly certain shortcomings in the day-to-day operations of safety procedures, but none had the least bearing on the disaster or its consequences and we do not take time with them.” Fortunately, others did not take this overly narrow viewpoint, and Flixborough led to major changes in the way hazardous facilities were allowed to operate in Britain.

Large-scale engineered systems are more than just a collection of technological artifacts: They are a reflection of the structure, management, procedures, and culture of the engineering organization that created them, and they are also, usually, a reflection of the society in which they were created. Ralph Miles Jr., in describing the basic concepts of systems theory, noted that:

> Underlying every technology is at least one basic science, although the technology may be well developed long before the science emerges. Overlying every technical or civil system is a social system that provides purpose, goals, and decision criteria. [78, p.1]

Effectively preventing accidents in complex systems requires using accident models that include that social system as well as the technology and its underlying science. Without understanding the purpose, goals, and decision criteria used to construct and operate systems, it is not possible to completely understand and most effectively prevent accidents.

Awareness of the importance of social and organizational aspects of safety goes back to the early days of system safety engineering. In 1968, Jerome Lederer, then the director of the NASA Manned Flight Safety Program for Apollo wrote:

> System safety covers the total spectrum of risk management. It goes *beyond the hardware* and associated procedures of system safety engineering. It involves: attitudes and motivation of designers and production people, employee/management rapport, the relation of industrial associations among themselves and with government, human factors

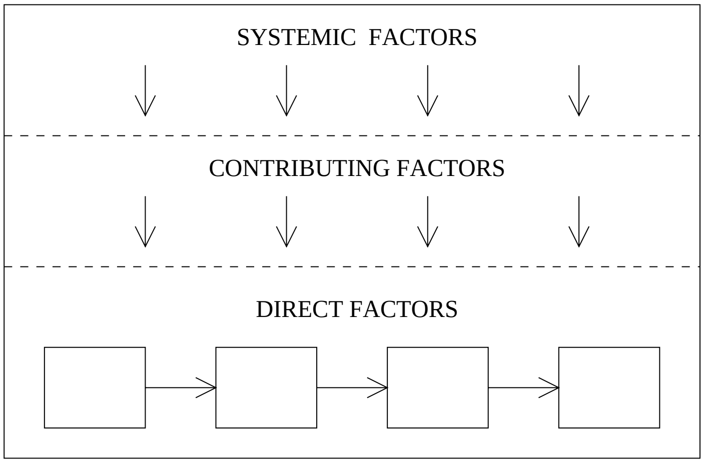


Figure 3.2: Johnson's Three Level Model of Accidents

> in supervision and quality control, documentation on the interfaces of industrial and public safety with design and operations, the interest and attitudes of top management, the effects of the legal system on accident investigations and exchange of information, the certification of critical workers, political considerations, resources, public sentiment and many other non-technical but vital influences on the attainment of an acceptable level of risk control. These non-technical aspects of system safety cannot be ignored [61].

Several attempts have been made to graft factors other than simple failure events and conditions onto event models, but all have been unsatisfactory, including my own previous attempts. The most common approach has been to add hierarchical levels above the event chain. In the seventies, Johnson proposed a model and sequencing method that described accidents as chains of direct events and causal factors arising from contributory factors, which in turn arise from systemic factors (Figure 3.2) [49].

Johnson also tried to put management factors into fault trees (a technique called MORT or Management Oversight Risk Tree), but ended up simply providing a general checklist for auditing management practices. While such a checklist can be very useful, it presupposes that every error can be predefined and put into a checklist form. The checklist is comprised of a set of questions that should be asked during an accident investigation. Examples of the questions from a DOE MORT User's Manual are: Was there sufficient training to update and improve needed supervisory skills? Did the supervisors have their own technical staff or access to such individuals? Was the technical support of the right discipline(s) sufficient for the needs of supervisory programs and review functions? Were there established methods for measuring performance that permitted the effectiveness of supervisory programs to be evaluated? Was a maintenance plan provided before startup? Was all relevant information provided to planners and managers? Was it used? Was concern for safety displayed by vigorous, visible personal action by top executives? etc. Johnson originally provided hundreds of such questions, and additions have been made to his checklist since Johnson created it in the 1970s so it is now even larger. The use of the MORT checklist is feasible because the items are so general, but that same generality also limits its usefulness.

In 1995, Leveson proposed a three-level model (based on an earlier conceptual model by Lewycky) [66] with slightly different types of information at the levels. In contrast to Johnson's model where the levels represent different types of factors, each level here provides a different model of the accident causal factors at three different levels of abstraction. The lowest level describes the accident mechanism in terms of an event chain. For example, an object appeared in front of the car, the driver hit the brake, the car skidded and hit the tree, the driver was thrown from the car and injured.

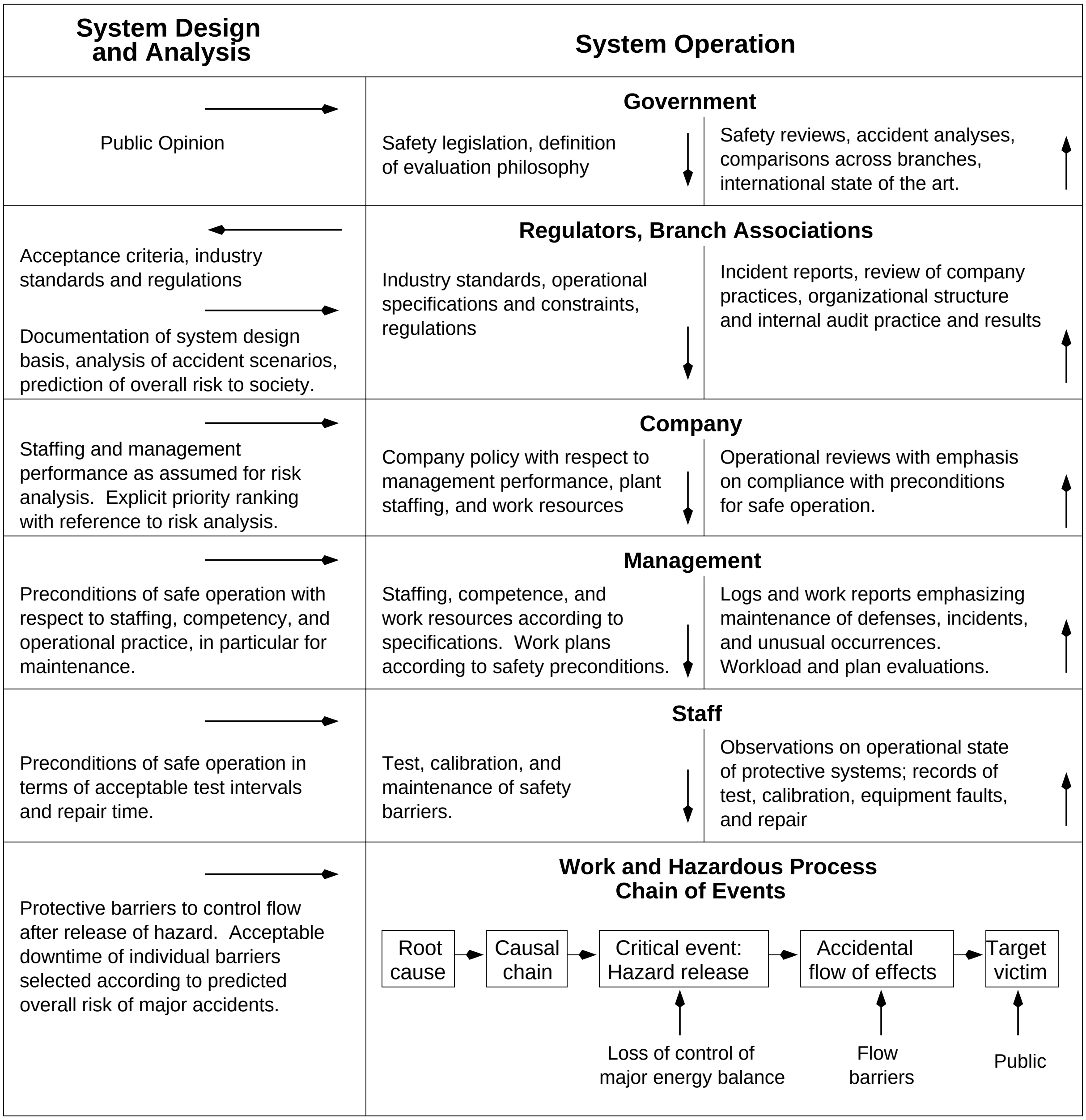
The second level is a model of the accident in terms of the conditions or lack of conditions that allowed the events at the first level to occur, e.g., the driver does not know how to prevent or stop the skid, the car is not equipped with anti-lock brakes, the driver was driving too fast, the street was wet from rain and thus friction was reduced, visibility was poor, the driver was not wearing a seat belt, the seat belt was defective. Systemic factors made up the third level, i.e., weaknesses of a technical, human, organizational, managerial, or societal nature that contributed to the accident, usually by allowing or causing the conditions to arise that led to the events.

The most sophisticated of these types of hierarchical add-ons to event chains is Rasmussen and Svedung's model of the socio-technical system involved in risk management [96]. As shown in Figure 3.3, at the social and organizational levels they use a hierarchical control structure, with levels for government, regulators and associations, company, management, and staff. At all levels they map information flow. The model concentrates on the operations component of the socio-technical system; information from the system design and analysis process is treated as input to the operations process. At each level, they model the factors involved using event chains, with links to the events chains at the level below. In addition, at the technical level they focus on the downstream part of the event chain following the occurrence of the hazard. This downstream emphasis is common in the process industry where Rasmussen has done most of his work. In such a downstream focus, emphasis is placed on protection or "safety systems" that identify a hazardous state after it occurs and then attempt to move the plant back into a safe state, often by means of a partial or total shutdown. While this type of shutdown design can work for process plants, it is not appropriate for all types of systems and suffers from a lack of emphasis on designing for safety and eliminating or controlling hazards in the basic system design.

One drawback to all these proposals is that the factors selected to be included are still arbitrary. An experimental application of Leveson's hierarchical model to eight aerospace accidents concluded that the model was an improvement over a basic chain of events model. Separation of the basic events from all the various types of explanations that could be given for those events allowed evaluation of the explanation in a more objective fashion and easier detection of omissions and biases. It also helped to identify conditions indirectly related to events or those related to all or a subset of the events. However, it (and the others that have been proposed) provide little help in selecting the events to include and, more important, in the selection of conditions and systemic factors.

Another drawback of such models is that they stick a hierarchy on top of the event chain without any real connection to it and with no way to show indirect relationships.

Most important, all the limitations of event chain models still exist, particularly at the technical or accident mechanism level. But event chains do not adequately handle the important new accident factors arising in the systems we are building today: system accidents caused by dysfunctional interactions among components and not individual component failure, complex human decision-making and supervisory control, software, and adaptation.



System Design and Analysis	System Operation	
Public Opinion	**Government** Safety legislation, definition of evaluation philosophy	Safety reviews, accident analyses, comparisons across branches, international state of the art.
Acceptance criteria, industry standards and regulations Documentation of system design basis, analysis of accident scenarios, prediction of overall risk to society.	**Regulators, Branch Associations** Industry standards, operational specifications and constraints, regulations	Incident reports, review of company practices, organizational structure and internal audit practice and results
Staffing and management performance as assumed for risk analysis. Explicit priority ranking with reference to risk analysis.	**Company** Company policy with respect to management performance, plant staffing, and work resources	Operational reviews with emphasis on compliance with preconditions for safe operation.
Preconditions of safe operation with respect to staffing, competency, and operational practice, in particular for maintenance.	**Management** Staffing, competence, and work resources according to specifications. Work plans according to safety preconditions.	Logs and work reports emphasizing maintenance of defenses, incidents, and unusual occurrences. Workload and plan evaluations.
Preconditions of safe operation in terms of acceptable test intervals and repair time.	**Staff** Test, calibration, and maintenance of safety barriers.	Observations on operational state of protective systems; records of test, calibration, equipment faults, and repair
Protective barriers to control flow after release of hazard. Acceptable downtime of individual barriers selected according to predicted overall risk of major accidents.	**Work and Hazardous Process Chain of Events**	

Root cause
Causal chain
Critical event: Hazard release
Accidental flow of effects
Target victim
Loss of control of major energy balance
Flow barriers
Public

Figure 3.3: The Ramussen/Svedung Model of Risk Management

3.2 System Accidents

Since World War II, we are increasingly experiencing a new type *system accident*[2] that arises in the interactions *among* components (electromechanical, digital, and human) rather than in the failure of individual components. In contrast, accidents arising from component failures, including the possibility of multiple and cascading failures, might be termed *component failure accidents.*

A *failure* in engineering can be defined as the nonperformance or inability of a component (or system) to perform its intended function. Intended function (and thus failure) is usually defined with respect to the component's behavioral requirements. If the behavior of a component satisfies its specified requirements, even though the requirements may include behavior that is undesirable from a larger system context, the component has not failed.

Accidents that do not involve component failure but instead result from *dysfunctional interactions* among system components have received less attention. This lack of concern may stem partly from the fact that in the simpler systems of the past, analysis and testing allowed exercising the system to detect all such potential interactions and changing the system design to eliminate them. Increasing complexity and the introduction of software control is reducing this ability and increasing the number of system accidents. System accidents can be explained in terms of inadequate control over component interactions. Prevention requires reducing or eliminating dysfunctional interactions, i.e., interactions that can lead to hazardous states in the controlled process. A taxonomy and classification of the types of dysfunctional interactions leading to accidents is shown in Section 7.2.

The Ariane 5 and Mars Polar Lander losses are examples of system accidents. In both of these accidents, the components did not fail in terms of not satisfying their specified requirements. The individual components operated exactly the way the designers had planned—the problems arose in the unplanned or misunderstood effects of these component behaviors on the system as a whole, that is, errors in the system design rather than the component design, including errors in allocating and tracing the system functions to the individual components. The solution, therefore, lies in systems engineering.

Consider an example of a system accident that occurred in a batch chemical reactor in England [56]. The design of this system is shown in Figure 3.4. The computer was responsible for controlling the flow of catalyst into the reactor and also the flow of water into the reflux condenser to cool off the reaction. Additionally, sensor inputs to the computer were supposed to warn of any problems in various parts of the plant. The programmers were told that if a fault occurred in the plant, they were to leave all controlled variables as they were and to sound an alarm.

On one occasion, the computer received a signal indicating a low oil level in a gearbox. The computer reacted as the requirements specified: It sounded an alarm and left the controls as they were. By coincidence, a catalyst had been added to the reactor, but the computer had just started to increase the cooling-water flow to the reflux condenser; the flow was therefore kept at a low rate. The reactor overheated, the relief valve lifted, and the content of the reactor was discharged into the atmosphere.

Note that there were no component failures involved in this accident: the individual components, including the software, worked as specified, but together they created a hazardous system state. Merely increasing the reliability of the individual components or protecting against their

[2] Perrow used this term to denote accidents resulting from failures of multiple components [87], but accidents occur even in systems where there have been no component failures.

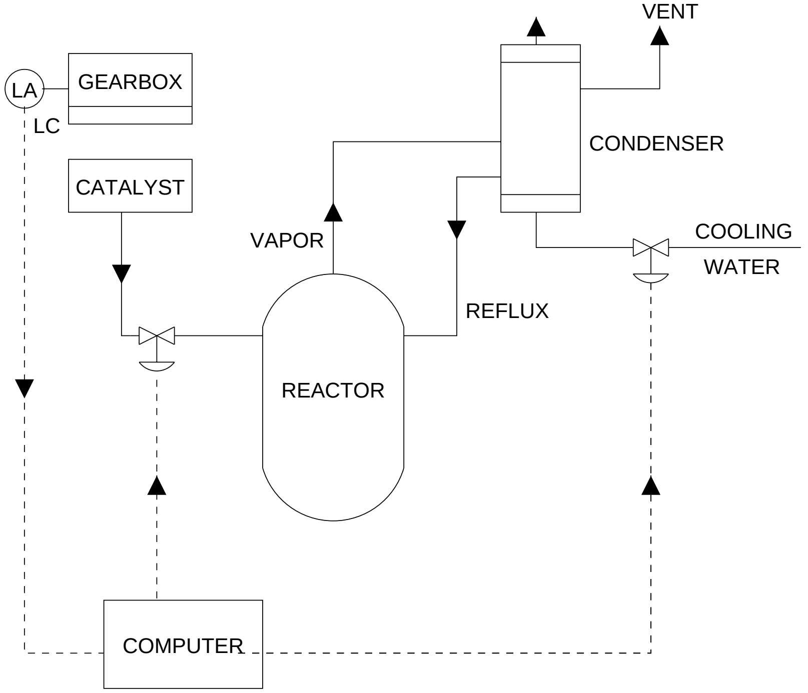


Figure 3.4: A chemical reactor design (adapted from Kletz [57, p.6]).

failure would not have prevented this accident. Prevention required identifying and eliminating or mitigating unsafe interactions among the system components.

While better engineering techniques aimed at increasing component integrity or reducing the effects of component failure on the system are reducing component failure accidents, system accidents are increasing in importance and will require new prevention approaches. In fact, some of the approaches typically used to reduce component failure accidents, such as redundancy, may actually increase the likelihood of system accidents by increasing one of the basic causal factors—interactive complexity.

The extraordinary interactive complexity of the systems we are trying to build and increased coupling between heterogeneous components (which may not be obvious or known) are two of the reasons behind the increase in system accidents. Both are related to the introduction of new technology, particularly software and digital technology.

While system design errors could usually be detected during system testing and fixed before use in pure physical systems, software is allowing us to build systems with such a high level of interactive complexity that potential interactions among components cannot be thoroughly planned, anticipated, tested, or guarded against. We are building systems that are intellectually unmanageable using currently available tools. In the long run, the solution is to extend our ability to manage complexity, but in the present we must deal with the tools we have and catching up to the continually increasing complexity of the systems we would like to build may be difficult.

A second factor implicated in system accidents is tight coupling. Loosely coupled or decoupled components and subsystems allow for intervention when problems arise or limit the ability of a disturbance in one part of the system to affect other parts. The drive toward tightly coupled systems is fueled by a desire for higher levels of efficiency and functionality, and the use of software allows us now to achieve levels of coupling and interaction that were previously impossible with pure electromechanical devices.

3.3 Human Error and Decision Making

Human error is usually defined as any deviation from the performance of a specified or prescribed sequence of actions. However, instructions and written procedures are almost never followed exactly as operators strive to become more efficient and productive and to deal with time pressures. In fact, a common way for workers to apply pressure to management without actually going out on strike is to *work to rule*, which can lead to a breakdown in productivity and even chaos.

In studies of operators, even in such highly constrained and high-risk environments as nuclear power plants, modification of instructions is repeatedly found and the violation of rules appears to be quite rational, given the actual workload and timing constraints under which the operators must do their job [37, 115, 127]. In these situations, a basic conflict exists between error viewed as a deviation from the *normative procedure* and error viewed as a deviation from the rational and normally used *effective procedure* [97].

One implication is that following an accident, it will be easy to find someone involved in the dynamic flow of events that has violated a formal rule by following *established practice* rather than *specified practice*. Given the frequent deviation of established practice from normative work instructions and rules, it is not surprising that operator "error" is found to be the cause of 70% to 80% of accidents. As noted in Section 2.2, a root cause is often selected because that event involves a deviation from a standard.

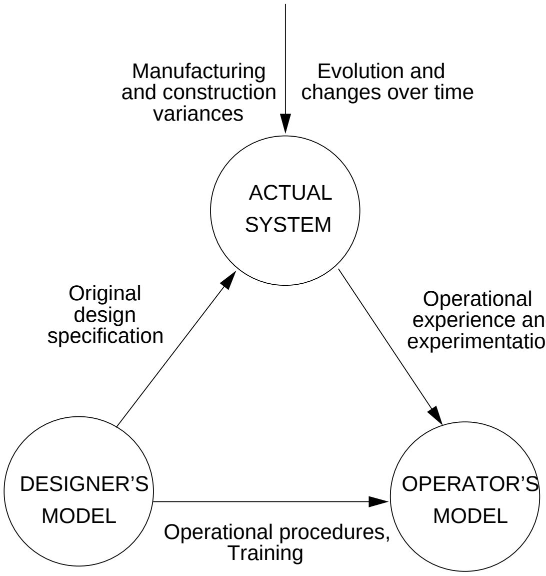


Figure 3.5: The relationship between mental models.

The updating of human mental models plays a significant role here (Figure 3.5). Both the designer and the operator will have their own mental models of the plant. It is quite natural for the designer's and operator's models to differ and even for both to have significant differences from the actual plant as it exists. During development, the designer evolves a model of the plant to the point where it can be built. The *designer's model* is an idealization formed *before* the plant is constructed. Significant differences may exist between this ideal model and the actual constructed system. Besides construction problems, the designer always deals with ideals or averages, not with the actual components themselves. Thus, a designer may have a model of a valve with an average closure time, while real valves have closure times that fall somewhere along a continuum of behavior that reflects manufacturing and material differences. The designer's model will be the basis for developing operator work instructions and training.

The *operator's model* will be based partly on formal training and partly on experience with the system. The operator must cope with the system as it is constructed and not as it may have been envisioned. In addition, physical systems will change over time and the operator's model must change accordingly. The only way for the operator to determine that the system has changed and that his or her mental model must be updated is through experimentation: To learn where the boundaries of safe behavior currently are, occasionally they must be crossed.

Experimentation is important at all levels of control [95]. For manual tasks where the optimization criteria are speed and smoothness, the limits of acceptable adaptation and optimization can only be known from the error experienced when occasionally crossing a limit. In this case, errors are an integral part of maintaining a skill at an optimal level and a necessary part of the feedback loop to achieve this goal. The role of such experimentation in accidents cannot be understood by treating human errors as events in a causal chain separate from the feedback loops in which they operate.

At higher levels of cognitive control, experimentation is needed for operators to update procedures to handle changing conditions or to evaluate hypotheses while engaged in symbolic reasoning about the best response to unexpected situations. Actions that are quite rational and important during the search for information and test of hypotheses may appear to be unacceptable mistakes in hindsight, without access to the many details of a "turbulent" situation [97].

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

Flawed decisions may also result from limitations in the boundaries of the model used, but the boundaries relevant to a particular decision maker may depend on activities of several other decision makers found within the total system [96]. Accidents may then result from the interaction of the potential side effects of the performance of the decision makers during their normal work. Before an accident, it will be difficult for the individual decision makers to see the total picture during their daily operational decision making and to judge the current state of the multiple defenses that are conditionally dependent on decisions taken by other people in other departments and organizations.

Using a ferry accident analyzed by Rasmussen and Svedung as an example (Figure 3.6), those making decisions about vessel design, harbor design, cargo management, passenger management, traffic scheduling, and vessel operation were unaware of the impact of their decisions on the others and the overall impact on the process leading to the ferry accident.

Rasmussen stresses that most decisions are sound using local judgment criteria and given the time and budget pressures and short-term incentives that shape behavior. Experts do their best to meet local conditions and in the busy daily flow of activities are unaware of the potentially dangerous side effects. Each individual decision may appear safe and rational within the context of the individual work environments and local pressures, but may be unsafe when considered as a whole: It is difficult if not impossible for any individual to judge the safety of their decisions when it is dependent on the decisions made by other people in other departments and organizations.

Traditional decision theory research perceives decisions as discrete processes that can be separated from the context and studied as an isolated phenomenon. More recent research has taken a very different approach: Instead of thinking of operations as predefined sequences of actions, human interaction with a system is increasingly being considered to be a continuous control task in which separate "decisions" or errors are difficult to identify.

Edwards, back in 1962, was one of the first to argue that decisions can only be understood as part of an ongoing process [33]. The state of the system is perceived in terms of possible actions, one of these actions is chosen, and the resulting response from the controlled system acts as a background for the next action. Errors then are difficult to localize in the stream of behavior; *the effects of less successful actions are a natural part of the search on the part of the operator for optimal performance.*

Not only are separate decisions difficult to identify in this model of human control, but the study of decision making then cannot be separated from a simultaneous study of the social context, the value system in which it takes place, and the dynamic work process it is intended to control [95]. This view is the foundation of *dynamic decision making* [15] and the new field of *naturalistic*

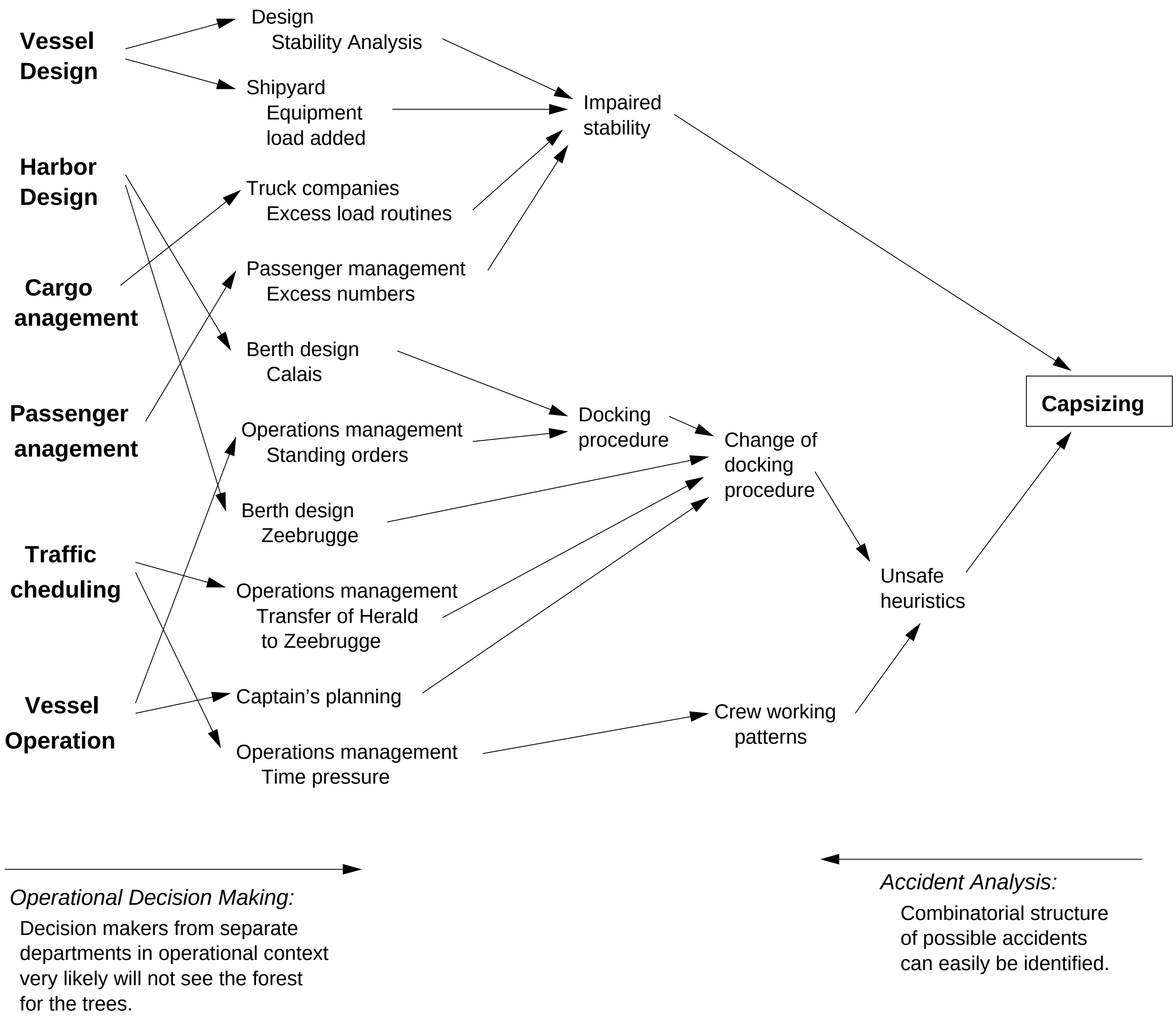

Vessel Design
Harbor Design
Cargo anagement
Passenger anagement
Traffic cheduling
Vessel Operation
Design Stability Analysis
Shipyard Equipment load added
Truck companies Excess load routines
Passenger management Excess numbers
Berth design Calais
Operations management Standing orders
Berth design Zeebrugge
Operations management Transfer of Herald to Zeebrugge
Captain's planning
Operations management Time pressure
Impaired stability
Docking procedure
Change of docking procedure
Crew working patterns
Unsafe heuristics
Capsizing
Operational Decision Making:
Decision makers from separate departments in operational context very likely will not see the forest for the trees.
Accident Analysis:
Combinatorial structure of possible accidents can easily be identified.

Figure 3.6: The complex interactions in the Zeebrugge accident (Adapted from Rasmussen [96, p.188]).

decision making [132, 55].

As argued by Rasmussen and others, devising more effective accident models requires shifting the emphasis in explaining the role of humans in accidents from error (deviations from normative procedures) to focus on the mechanisms and factors that shape human behavior, i.e., the performance-shaping mechanisms and context in which human actions take place and decisions are made. Modeling behavior by decomposing it into decisions and actions and studying it as a phenomenon isolated from the context in which the behavior takes place is not an effective way to understand behavior [96]. The alternative view requires a new approach to representing human behavior, focused not on human error and violation of rules but on the mechanisms generating behavior in the actual, dynamic context. Such an approach must take into account the work system constraints, the boundaries of acceptable performance, the need for experimentation, and subjective criteria guiding adaptation to change. In this approach, traditional task analysis is replaced or augmented with *cognitive work analysis* [97, 116] or *cognitive task analysis* [40].

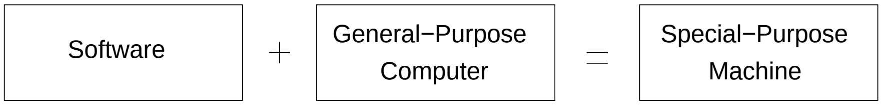
To accomplish this goal, a representation of human behavior at a higher level of functional abstraction than the level used for task analysis is needed. Such a representation must include the objectives of the individual actor in the actual context and the boundaries of acceptable performance, that is, the constraints on the work space. Rasmussen, Pejtersen, and Goodstein [97] define one such framework for identifying the objectives, value structures, and subjective preferences governing behavior within the degrees of freedom faced by the individual decision maker and actor. Vicente also uses this framework for his Cognitive Work Analysis [116]. In this approach, behavior is modeled directly in terms of the behavior-shaping constraints of the environment and the adaptive mechanisms of human actors in the environment.

Using this view of human error leads to a new approach to control of human performance: Rather than trying to control behavior by fighting deviations from a particular path, focus should be on control of behavior by identifying the boundaries of safe performance, by making boundaries explicit and known, by giving opportunities to develop coping skills at the boundaries, by designing systems to support safe optimization and adaptation of performance in response to contextual influences and pressures, by providing means for identifying potentially dangerous side effects of individual decisions in the network of decisions over the entire system, by designing for error tolerance (making errors observable and reversible before safety constraints are violated), and by counteracting the pressures that drive operators and decision makers to violate safety constraints.

3.4 Software Error

One of the common factors involved in system accidents is the use of software control. Software and digital automation introduces a new factor into the engineering of complex systems. This new factor requires changes to existing systems engineering techniques and places new requirements on accident models.

The uniqueness and power of the digital computer over other machines stems from the fact that, for the first time, we have a general-purpose machine:

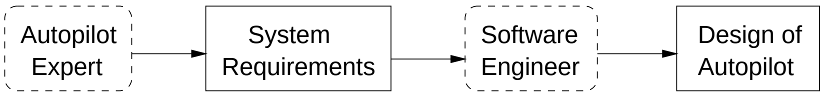


We no longer need to build a mechanical or analog autopilot from scratch, for example, but simply to write down the "design" of an autopilot in the form of instructions or steps to accomplish the desired goals. These steps are then loaded into the computer, which, while executing the instructions, in effect *becomes* the special-purpose machine (the autopilot). If changes are needed, the instructions can be changed instead of building a different physical machine from scratch. Software in essence is the *design of a machine abstracted from its physical realization.*

Machines that previously were physically impossible or impractical to build become feasible, and the design of a machine can be changed quickly without going through an entire retooling and manufacturing process. In essence, the manufacturing phase is eliminated from the lifecycle of these machines: The physical parts of the machine (the computer hardware) can be reused, leaving only the design and verification phases. The design phase also has changed: Emphasis is placed only on the steps to be achieved without having to worry about how those steps will be realized physically.

These advantages of using computers (along with others specific to particular applications, such as reduced size and weight) have led to an explosive increase in their use, including their introduction into potentially dangerous systems. There are, however, some potential disadvantages of using computers and some important changes that their use introduces into the traditional engineering process that are leading to new types of accidents as well as creating difficulties in investigating accidents and preventing them.

With computers, the design of a machine is usually created by someone who is not an expert in its design. The autopilot design expert, for example, decides how the autopilot should work, but then provides that information to a software engineer who is an expert in software design but not autopilots. It is the software engineer who then creates the detailed design of the autopilot. The extra communication step between the engineer and the software developer is the source of the most serious problems with software today.



It should not be surprising then that most errors found in operational software can be traced to requirements flaws, particularly incompleteness[3]. In addition, nearly all the serious accidents in which software has been involved in the past 20 years can be traced to requirements flaws, not coding errors. The requirements may reflect incomplete or wrong assumptions about the operation of the system components being controlled by the software or about the operation of the computer itself. The problems may also stem from unhandled controlled-system states and environmental conditions. Thus simply trying to get the software "correct" in terms of accurately implementing the requirements will not make it safer in most cases. Software may be highly reliable and correct and still be unsafe when:

- The software correctly implements the requirements but the specified behavior is unsafe from a system perspective;

[3] Completeness is a quality often associated with requirements but rarely defined. The most appropriate definition in the context of this book has been proposed by Jaffe: Software requirements specifications are complete if they are sufficient to distinguish the desired behavior of the software from that of any other undesired program that might be designed [48].

- The software requirements do not specify some particular behavior required for system safety (that is, they are incomplete);

- The software has unintended (and unsafe) behavior beyond what is specified in the requirements.

Basically the problems stem from the software doing what the software engineer thought it should do when that is not what the original design engineer wanted. Integrated product teams and other project management schemes to help with this communication are being used, but the problem has not been adequately solved as witnessed by the increasing number of system accidents related to these types of communication problems.

Even if the requirements problem was solved, there are other important problems that the use of software introduces into the system safety equation. First, the "failure modes" of software differ from physical devices. Software is simply the design of the machine abstracted from its physical realization; for example, the logical design of an autopilot independent of any concurrent physical design to realize that logic in hardware. Can software "fail"? What does it mean for a design to fail? Obviously, if the term failure has any meaning whatsoever in this context, it has to be different than that implied by the failure of a physical device. Most software-related accidents stem from the operation of the software, not from its *lack* of operation and usually that operation is exactly what the software engineers intended. Thus event-based accident models as well as reliability analysis methods that focus on classic types of failure events will not apply to software. Confusion about this point is reflected in the many fault trees containing boxes that say "*Software fails.*"

The third problem can be called the *curse of flexibility.* The computer is so powerful and so useful because it has eliminated many of the physical constraints of previous machines. This is both its blessing and its curse: We no longer have to worry about the physical realization of our designs, but we also no longer have physical laws that limit the complexity of our designs. Physical constraints enforce discipline on the design, construction, and modification of our design artifacts. Physical constraints also control the complexity of what we build. With software, the limits of what is *possible* to accomplish are different than the limits of what can be accomplished *successfully* and *safely*—the limiting factors change from the structural integrity and physical constraints of our materials to limits on our intellectual capabilities. It is possible and even quite easy to build software that we cannot understand in terms of being able to determine how it will behave under all conditions. We can construct software (and often do) that goes beyond human intellectual limits. The result has been an increase in *system accidents* stemming from intellectual unmanageability related to interactively complex and tightly coupled designs that allow potentially unsafe interactions to go undetected during development.

One possible solution is to stretch our intellectual limits by using mathematical modeling and analysis. Engineers make extensive use of models to understand and predict the behavior of physical devices. Although computer scientists realized that software could be treated as a mathematical object over 30 years ago, mathematical methods (called *formal* methods in computer science) have not been widely used on software in industry, although there have been some successes in using them on computer hardware. There are several reasons for this lack of use. The biggest problem is simply the complexity of such models. Software has a very large number of states (a model we created of the function computed by TCAS II, a collision avoidance system required on most commercial aircraft in the U.S., has upwards of 10^{40} states). Sheer numbers would not be a problem if the states exhibited adequate regularity to allow reduction in the complexity based on grouping and equivalence classes. Unfortunately, application software does not exhibit the same type of regularity found in digital hardware.

Physical systems, of course, also have a large number of states (in fact, often infinite) but physical continuity allows the use of continuous math, where one equation can describe an infinite number of states. Software lacks that physical continuity, and discrete math must be used. In place of continuous math, such as differential equations, formal logic is commonly used to describe the required characteristics of the software behavior. However, specifications using formal logic may be the same size or even larger than the code, more difficult to construct than the code, and harder to understand than the code. Therefore, they are as difficult and error-prone to construct, if not more so, than the software itself. That doesn't mean they cannot be useful, but they are not going to be a panacea for our problems.

Testing, while necessary, is not the solution to the problem either. The enormous number of states in most software means that only a very small percentage can be tested and the lack of continuity does not allow for sampling and interpolation of behavior between the sampled states. For most nontrivial software, exhaustive testing to cover all paths through the logic would take centuries.

One limitation in the current formal models and tools is that most require a level of expertise in discrete math that is commonly attained only by those with advanced degrees in computer science or applied math. More serious is the problem that the models do not match the way engineers usually think about their designs and therefore are difficult to review and validate. The basic problems will not be solved by providing tools to evaluate the required behavior of the special-purpose machine being designed that can be used only by computer scientists. Instead, tools are needed that are usable by the expert in the special-purpose machine's design to specify and evaluate the behavior. At the same time, software engineers *do* need models and tools to analyze the structural design of the software itself. Thus, for each group, different tools and models are required.

One additional complication is simply the number of emergent properties exhibited by software. Like most complex designs, errors are more likely to be found in the interactions among the software components than in the design of the individual components. Emergent properties complicate the creation of effective analysis methods.

In summary, accident models and system safety approaches will be ineffective for software-intensive systems unless they can handle the unique characteristics of software. They will need to account for the abstract nature of software and the different role it plays in accidents.

3.5 Adaptation

Any accident model that includes the social system and humans must account for adaptation. To paraphrase a familiar saying, the only constant is that nothing ever remains constant. Systems and organizations continually experience change as adaptations are made in response to local pressures and short-term productivity and cost goals. People adapt to their environment or they change their environment to better suit their purposes. A corollary of this propensity for systems and people to adapt over time is that safety defenses are likely to degenerate systematically through time, particularly when pressure toward cost-effectiveness and increased productivity is the dominant element in decision making. The critical factor here is that such adaptation is not a random process—it is an optimization process depending on search strategies—and thus should be predictable and potentially controllable [96].

Woods has stressed the importance of adaptation in accidents. He describes organizational and human failures as breakdowns in adaptations directed at coping with complexity and accidents as involving a "drift toward failure as planned defenses erode in the face of production pressures and change" [129].

Similarly, Rasmussen has argued that major accidents are often caused not by a coincidence of independent failures but instead reflect a systematic migration of organizational behavior to the boundaries of safe behavior under pressure toward cost-effectiveness in an aggressive, competitive environment [96]. One implication of this viewpoint is that the struggle for a good safety culture will never end because it must continually fight against the functional pressures of the work environment. Improvement of the safety culture will therefore require an analytical approach directed toward the behavior-shaping factors in the environment.

Humans and organizations can adapt and still maintain safety as long as they stay within the area bounded by safety constraints. But in the search for optimal operations, humans and organizations will close in on and explore the boundaries of established practice, and such exploration implies the risk of occasionally crossing the limits of safe practice unless the constraints on safe behavior are enforced.

The natural migration toward the boundaries of safe behavior, according to Rasmussen, is complicated by the fact that it results from the decisions of multiple people, in different work environments and contexts within the overall socio-technical system, all subject to competitive or budgetary stresses and each trying to optimize their decisions within their own immediate context. Several decision makers at different times, in different parts of the company or organization, all striving locally to optimize cost effectiveness may be preparing the stage for an accident, as illustrated by the Zeebrugge ferry accident (see Figure 3.6) and the friendly fire accident described in Chapter 7. The dynamic flow of events can then be released by a single act.

For an accident model to handle system adaptation over time, it must consider the processes involved in accidents and not simply events and conditions: Processes control a sequence of events and describe system and human behavior as it changes and adapts over time rather than considering individual events and human actions. To talk about *the cause* or *causes* of an accident makes no sense in this view of accidents. As Rasmusssen argues, deterministic, causal models are inadequate to explain the organizational and social factors in highly adaptive socio-technical systems. Instead, accident causation must be viewed as a complex *process* involving the entire socio-technical system including legislators, government agencies, industry associations and insurance companies, company management, technical and engineering personnel, operations, etc.

3.6 Goals for a New Accident Model

Event-based models work best for accidents where one or several components fail, leading to a system failure or hazard. However, accident models and explanations involving only simple chains of failure events can easily miss subtle and complex couplings and interactions among failure events and omit entirely accidents involving no component failure at all. The event-based models developed to explain physical phenomena (which they do well) are inadequate to explain accidents involving organizational and social factors and human decisions and software design errors in highly adaptive, tightly-coupled, interactively complex socio-technical systems—namely, those accidents related to the new factors (described in Chapter 1) in the changing environment in which engineering is taking place.

The search for a new model, resulting in the accident model presented in Part II of this book, was driven by the following goals:

- *Expanding accident analysis by forcing consideration of factors other than component failures and human errors.* The model should encourage a broad view of accident mechanisms, expanding the investigation from simply considering proximal events to considering the entire socio-technical system. Such a model should include societal, regulatory, and cultural factors. While some accident reports do this well, for example the space shuttle *Challenger* report, such results appear to be ad hoc and dependent on the personalities involved in the investigation rather than being guided by the accident model itself.

- *Providing a more scientific way to model accidents that produces a better and less subjective understanding of why the accident occurred and how to prevent future ones.* Event chain models provide little guidance in the selection of events to include in the accident explanation or the conditions to investigate. The model should provide more assistance in identifying and understanding a comprehensive set of factors involved and in identifying the adaptations that led to the loss.

- *Including system design errors and dysfunctional system interactions.* The models used widely were created before computers and digital components and do not handle them well. In fact, many of the event-based models were developed to explain industrial accidents, such as workers falling into holes or injuring themselves during the manufacturing process, and do not fit system safety at all. A new model must be able to account for accidents arising from dysfunctional interactions among the system components.

- *Allowing for and encouraging new types of hazard analyses and risk assessments that go beyond component failures and can deal with the complex role software and humans are assuming in high-tech systems.* Traditional hazard analysis techniques, such as fault tree analysis and the various types of failure analysis techniques, do not work well for software and other system design errors. An appropriate model should suggest hazard analysis techniques to augment these failure-based methods and encourage a wider variety of risk reduction measures than redundancy and monitoring. Risk assessment is currently firmly rooted in the probabilistic analysis of failure events. Attempts to extend current probabilistic risk assessment techniques to software and other new technology, to management, and to cognitively complex human control activities have been disappointing. This way forward may lead to a dead end, but starting from a different theoretical foundation may allow significant progress in finding new, more comprehensive approaches to risk assessment for complex systems.

- *Shifting the emphasis in the role of humans in accidents from errors (deviations from normative behavior) to focus on the mechanisms and factors that shape human behavior (i.e., the performance-shaping mechanisms and context in which human actions take place and decisions are made).* A new model should account for the complex role that human decisions and behavior are playing in the accidents occurring in high-tech systems and handle not simply individual decisions but sequences of decisions and the interactions among decisions by multiple, interacting decision makers [96]. The model must include examining the possible goals and motives behind human behavior.

- *Encouraging a shift in the emphasis in accident analysis from "cause"—which has a limiting, blame orientation—to understanding accidents in terms of reasons, i.e., why the events and errors occurred [112].* Learning how to engineer safer systems is the goal here, not identifying who to punish.

- *Examining the processes involved in accidents and not simply events and conditions.* Processes control a sequence of events and describe changes and adaptations over time rather than considering events and human actions individually.

- *Allowing for and encouraging multiple viewpoints and multiple interpretations when appropriate.* Operators, managers, and regulatory agencies may all have different views of the flawed processes underlying an accident, depending on the hierarchical level of the socio-technical control structure from which the process is viewed. At the same time, the factual data should be separated from the interpretation of that data.

- *Assisting in defining operational metrics and analyzing performance data.* Computers allow the collection of massive amounts of data, but analyzing that data to determine whether the system is moving toward the boundaries of safe behavior is difficult. A new accident model should provide directions for identifying appropriate safety metrics and operational auditing procedures to evaluate decisions made during design and development, to determine whether controls over hazards are adequate, to detect erroneous operational and environmental assumptions underlying the hazard analysis and design process, to identify dangerous trends and changes in operations before they lead to accidents, and to identify any maladaptive system or environment changes over time that could increase risk of accidents to unacceptable levels.

These goals are achievable if models based on systems theory underlie our safety engineering activities.

Chapter 4

Systems Theory and its Relationship to Safety

4.1 An Introduction to Systems Theory

Systems theory dates from the thirties and forties and was a response to limitations of the classic analysis techniques in coping with the increasingly complex systems being built [18]. Norbert Weiner applied the approach to control and communications engineering while Ludwig von Bertalanffy developed similar ideas for biology. It was Bertalanffy who suggested that the emerging ideas in various fields could be combined into a general theory of systems.

In the traditional scientific method, sometimes referred to as *divide and conquer*, systems are broken into distinct parts so that the parts can be examined separately: Physical aspects of systems are decomposed into separate physical components while behavior is decomposed into events over time.

Such decomposition (formally called *analytic reduction*) assumes that such separation is feasible: that is, each component or subsystem operates independently and analysis results are not distorted when these components are considered separately. This assumption in turn implies that the components or events are not subject to feedback loops and other non-linear interactions and that the behavior of the components is the same when examined singly as when they are playing their part in the whole. A third fundamental assumption is that the principles governing the assembling of the components into the whole are straightforward, that is, the interactions among the subsystems are simple enough that they can be considered separate from the behavior of the subsystems themselves.

These are reasonable assumptions, it turns out, for many of the physical regularities of the universe. System theorists have described these systems as displaying *organized simplicity* (Figure 4.1) [120]. Such systems can be separated into non-interacting subsystems for analysis purposes: the precise nature of the component interactions is known and interactions can be examined pairwise. This approach has been highly effective in physics and is embodied in structural mechanics.

Other types of systems display what systems theorists have labeled *unorganized complexity*—that is, they lack the underlying structure that allows reductionism to be effective. They can, however, often be treated as aggregates: They are complex but regular and random enough in their behavior that they can be studied statistically. This study is simplified by treating them as a

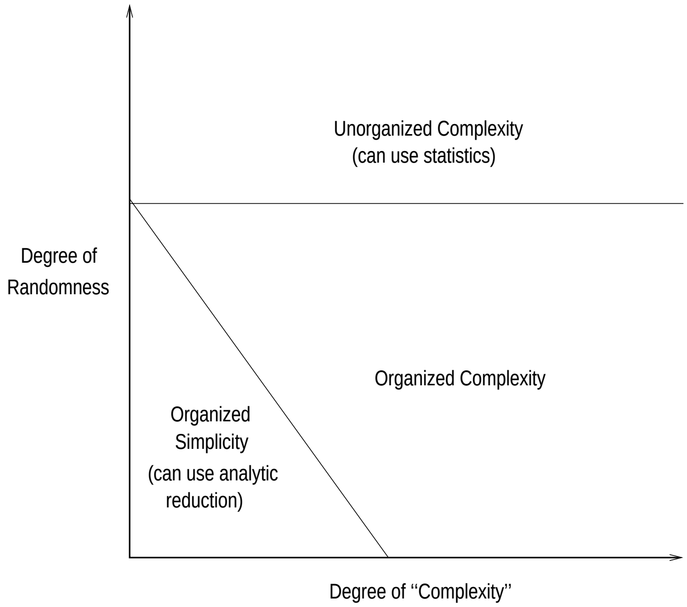


Figure 4.1: Three Categories of Systems (Adapted from Gerald Weinberg, *An Introduction to General Systems Thinking*, John Wiley, 1975)

structureless mass with interchangeable parts and then describing them in terms of averages. The basis of this approach is the *law of large numbers*: The larger the population, the more likely that observed values are close to the predicted average values. In physics, this approach is embodied in statistical mechanics.

A third type of system exhibits what system theorists call *organized complexity*: These systems are too complex for complete analysis and too organized for statistics; the averages are deranged by the underlying structure [120]. It is this type of system, which describes many of the complex engineered systems of the post-World War II era as well as biological systems and social systems, that is the subject of systems theory. Organized complexity also represents particularly well the problems that are faced by those attempting to build complex software, and it explains the difficulty computer scientists have had in attempting to apply analysis and statistics to such software. In recent years, more computer scientists are recognizing the limitations and are looking to systems theory. In fact, many of the most successful forms of abstraction developed and used by computer scientists from the very beginnings of the field embody basic systems theory principles.

The systems approach focuses on systems taken as a whole, not on the parts taken separately. It assumes that some properties of systems can only be treated adequately in their entirety, taking into account all facets relating the social to the technical aspects [91]. These system properties derive from the relationships between the parts of systems: how the parts interact and fit together [2]. Thus, the system approach concentrates on the analysis and design of the whole as distinct from the components or the parts and provides a means for studying systems exhibiting organized complexity.

The foundation of systems theory rests on two pairs of ideas: (1) *emergence* and *hierarchy* and (2) *communication* and *control* [18].

4.2 Emergence and Hierarchy

A general model of complex systems can be expressed in terms of a *hierarchy* of levels of organization, each more complex than the one below, where a level is characterized by having *emergent* properties. Emergent properties do not exist at lower levels; they are meaningless in the language appropriate to those levels. The shape of an apple, although eventually explainable in terms of the cells of the apple, has no meaning at that lower level of description. Thus, the operation of the processes at the lower levels of the hierarchy result in a higher level of complexity—that of the whole apple itself—that has emergent properties, one of them being the apple's shape [18]. The concept of emergence is the idea that at a given level of complexity, some properties characteristic of that level (emergent at that level) are irreducible. In computer science, deadlock is an example of such a property.

Hierarchy theory deals with the fundamental differences between one level of complexity and another. Its ultimate aim is to explain the relationships between different levels: what generates the levels, what separates them, and what links them. Emergent properties associated with a set of components at one level in a hierarchy are related to *constraints upon the degree of freedom* of those components. Describing the emergent properties resulting from the imposition of constraints requires a language at a higher level (a metalevel) different than that describing the components themselves. Thus, different languages of description are appropriate at different levels.

Safety is clearly an emergent property of systems. Determining whether a plant is acceptably safe is not possible by examining a single valve in the plant. In fact, statements about the "safety of the valve" without information about the context in which that valve is used, are meaningless. Conclusions can be reached, however, about the reliability of the valve, where reliability is defined as the probability that the behavior of the valve will satisfy its specification over time and under given conditions. This is one of the basic distinctions between safety and reliability: Safety can only be determined by the relationship between the valve and the other plant components—that is, in the context of the whole. Therefore it is not possible to take a single system component, like a software module, in isolation and assess its safety. A component that is perfectly safe in one system may not be when used in another.

The new model of accidents introduced in Part II of this book incorporates the basic systems theory idea of hierarchical levels, where constraints or lack of constraints at the higher levels control or allow lower-level behavior. Safety is treated as an emergent property at each of these levels.

4.3 Communication and Control

The second major pair of ideas in systems theory is *communication* and *control.* An example of regulatory or *control* action is the imposition of *constraints* upon the activity at one level of a hierarchy, which define the "laws of behavior" at that level yielding activity meaningful at a higher level. Hierarchies are characterized by control processes operating at the interfaces between levels [18]. The link between control mechanisms studied in natural systems and those engineered in man-made systems was provided by a part of systems theory known as cybernetics. Checkland writes:

> Control is always associated with the imposition of constraints, and an account of a control process necessarily requires our taking into account at least two hierarchical

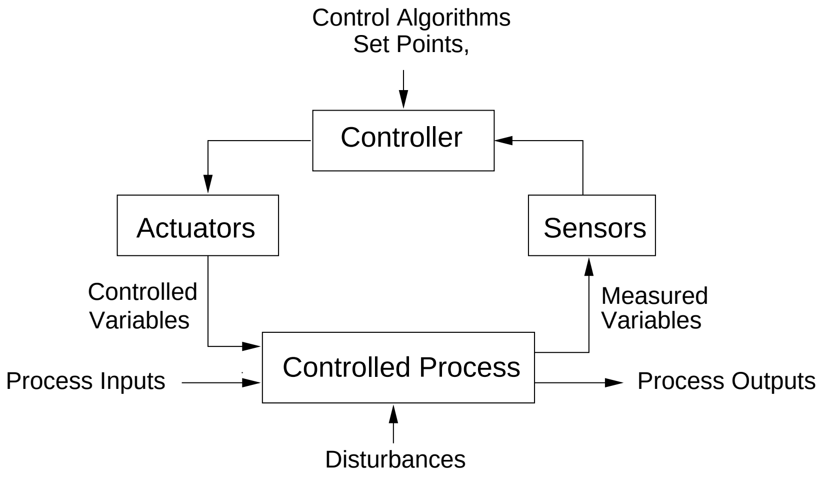


Figure 4.2: A standard control loop.

> levels. At a given level, it is often possible to describe the level by writing dynamical equations, on the assumption that one particle is representative of the collection and that the forces at other levels do not interfere. But any description of a control process entails an upper level imposing constraints upon the lower. The upper level is a source of an alternative (simpler) description of the lower level in terms of specific functions that are emergent as a result of the imposition of constraints [18, p.87].

Control in open systems (those that have inputs and outputs from their environment) implies the need for *communication*. Bertalanffy distinguished between *closed systems*, in which unchanging components settle into a state of equilibrium, and *open systems*, which can be thrown out of equilibrium by exchanges with their environment.

In control theory, open systems are viewed as interrelated components that are kept in a state of dynamic equilibrium by feedback loops of information and control. The plant's overall performance has to be controlled in order to produce the desired product while satisfying cost and quality constraints. In general, to effect control over a system requires four conditions [5]:

- **Goal Condition**: The controller must have a goal or goals (e.g., to maintain the setpoint).
- **Action Condition**: The controller must be able to affect the state of the system.
- **Model Condition**: The controller must be (or contain) a model of the system (see Section 7.1).
- **Observability Condition**: The controller must be able to ascertain the state of the system.

Figure 4.2 shows a typical control loop. The plant controller obtains information about (observes) the process state from measured variables (*feedback*) and uses this information to initiate action by manipulating *controlled variables* to keep the process operating within predefined limits or *set points* (the goal) despite disturbances to the process. In general, the maintenance of any open-system hierarchy (either biological or man-made) will require a set of processes in which there is communication of information for regulation or control [18].

Control actions will, in general, lag in their effects on the process because of delays in signal propagation around the control loop: an actuator may not respond immediately to an external command signal (called *dead time*); the process may have delays in responding to manipulated variables (*time constants*); and the sensors may obtain values only at certain sampling intervals (*feedback delays*). Time lags restrict the speed and extent with which the effects of disturbances, both within the process itself and externally derived, can be reduced. They also impose extra requirements on the controller, for example, the need to infer delays that are not directly observable.

4.4 Using System Theory to Understand Accidents

Models based on system theory consider accidents as arising from the interactions among system components and usually do not specify single causal variables or factors [63]. Whereas industrial (occupational) safety models focus on unsafe acts or conditions, classic system safety models instead look at what went wrong with the system's operation or organization to allow the accident to take place.

This systems approach treats safety as an emergent property that arises when the system components interact within an environment. Emergent properties are controlled or enforced by a set of constraints (control laws) related to the behavior of the system components. Accidents result from interactions among components that violate these constraints—in other words, from a lack of appropriate constraints on the interactions.

Safety then can be viewed as a control problem. Accidents occur when component failures, external disturbances, and/or dysfunctional interactions among system components are not adequately handled by the control system. In the space shuttle *Challenger* accident, for example, the O-rings did not adequately control propellant gas release by sealing a tiny gap in the field joint. In the Mars Polar Lander loss, the software did not adequately control the descent speed of the spacecraft—it misinterpreted noise from a Hall effect sensor (feedback of a measured variable) as an indication the spacecraft had reached the surface of the planet. Accidents such as these, involving engineering design errors, may in turn stem from inadequate control over the development process. Control is also imposed by the management functions in an organization—the *Challenger* and Columbia accidents involved inadequate controls in the launch-decision process, for example.

While events reflect the *effects* of dysfunctional interactions and inadequate enforcement of safety constraints, the inadequate control itself is only indirectly reflected by the events—the events are the *result* of the inadequate control. The control structure itself must be examined to determine why it was inadequate to maintain the constraints on safe behavior and why the events occurred.

As an example, the unsafe behavior (hazard) in the *Challenger* loss was the release of hot propellant gases from the field joint. The miscreant O-ring was used to control the hazard, i.e., its role was to seal a tiny gap in the field joint created by pressure at ignition. The loss occurred because the system design did not effectively impose the required constraint on the propellant gas release. Starting from here, there are then several questions that need to be answered to understand why the accident occurred and to obtain the information necessary to prevent future accidents. Why was this particular design unsuccessful in imposing the constraint, why was it chosen (what was the decision process), why was the flaw not found during development, and was there a different design that might have been more successful? These questions and others consider the original *design process.*

Understanding the accident also requires examining the contribution of the *operations process.* One constraint that was violated during operations was the requirement to correctly handle feedback about any potential violation of the safety design constraints, in this case, feedback during operations that the control by the O-rings of the release of hot propellant gases from the field joints was not being adequately enforced by the design. There were several instances of feedback that was not adequately handled, such as data about O-ring blowby and erosion during previous shuttle launches and feedback by engineers who were concerned about the behavior of the O-rings in cold weather. In addition, there was missing feedback about changes in the design and testing procedures during operations, such as the use of a new type of putty and the introduction of new O-ring leak checks without adequate verification that they satisfied system safety constraints on the field joints. As a final example, the control processes that ensured unresolved safety concerns were adequately considered before each flight, i.e., flight readiness reviews and other feedback channels to project management making flight decisions, were flawed.

Systems theory provides a much better foundation for safety engineering than the classic analytic reduction approach underlying event-based models of accidents. In addition, system safety fits naturally within the field of systems engineering: both rest on the same theoretical foundations and require engineering the system as a whole.

4.5 Systems Engineering and Safety

The emerging theory of systems, along with many of the historical forces noted in Chapter 1, gave rise after World War II to a new emphasis in engineering, eventually called systems engineering. During and after the war, technology expanded rapidly and engineers were faced with designing and building more complex systems than had been attempted previously. Much of the impetus for the creation of this new discipline came from military programs in the 1950s and 1960s, particularly ICBM systems. *Apollo* was the first nonmilitary government program in which systems engineering was recognized from the beginning as an essential function [13]. System safety was created at the same time and for the same reasons and is often considered to be a subdiscipline of systems engineering.

Systems theory provided the theoretical foundation for systems engineering, which views each system as an integrated whole even though it is composed of diverse, specialized components. The objective is to integrate the subsystems into the most effective system possible to achieve the overall objectives, given a prioritized set of design criteria. Optimizing the system design often requires making tradeoffs between these design criteria (goals).

Although many of the techniques of systems engineering were developed and were being used before the creation of the formal discipline and the underlying theory, the development of systems engineering as a discipline enabled the solution of enormously more complex and difficult technological problems than before [78]. Many of the elements of systems engineering can be viewed merely as good engineering: It represents more a shift in emphasis than a change in content. In addition, while much of engineering is based on technology and science, systems engineering is equally concerned with overall management of the engineering process.

A systems engineering approach to safety starts with the basic assumption that some properties of systems, in this case safety, can only be treated adequately in their entirety, taking into account all variables and relating the social to the technical aspects [91]. A basic assumption of systems engineering is that optimization of individual components or subsystems will not in general lead to a system optimum; in fact, improvement of a particular subsystem may actually worsen the overall system performance because of complex, non-linear interactions among the components. Similarly, individual component behavior (including events or actions) cannot be understood without considering the components' role and interaction within the system as a whole. This basis for systems engineering has been stated as the principle that a system is more than the sum of its parts. Attempts to improve long-term safety in complex systems by analyzing and changing individual components have often proven to be unsuccessful over the long term. For example, Rasmussen notes that over many years of working in the field of nuclear power plant safety, he found that attempts to improve safety from models of local features were compensated for by people adapting to the change in an unpredicted way [96].

Any accident model based on systems theory must take these basic systems engineering principles into account. Models that do not account for them will be limited in the types of accidents and systems they can handle. At the same time, models that include them have the potential to greatly improve our ability to engineer safer and more complex systems having greater interactive complexity and coupling.

4.6 Building Safety into the System Design

Safety fits naturally within the general systems engineering process and the problem-solving approach that a system view provides. This problem-solving process entails several steps (Figure 4.3). First, a need or problem is specified in terms of objectives that the system must satisfy along with criteria that can be used to rank alternative designs. Then, a process of system synthesis takes place that results in a set of alternative designs. Each of these alternatives is analyzed and evaluated in terms of the stated objectives and design criteria, and one alternative is selected to be implemented. In practice, the process is highly iterative: The results from later stages are fed back to early stages to modify objectives, criteria, design alternatives, and so on.

System safety should be treated as an integral component of systems engineering, as is common in the defense industry (and defined in MIL-STD-882). The following are some examples of basic systems engineering activities and the role of safety within them:

- *Needs analysis:* The starting point of any system design project is a perceived need. This need must first be established with enough confidence to justify the commitment of resources to satisfy it and understood well enough to allow appropriate solutions to be generated. Design criteria must be established to provide a means to evaluate both the evolving and final system. These criteria will include criteria for a safe design if safety is a perceived need for the project, i.e., if there are hazards associated with the operation of the system.

- *Feasibility studies:* The goal of this step in the design process is to generate a set of realistic designs. This goal is accomplished by (1) identifying the principal constraints—including safety constraints—and design criteria for the specific problem being addressed; (2) generating plausible solutions to the problem that satisfy the requirements and constraints; and (3) selecting potential solutions on the basis of physical and economic feasibility.

- *Trade studies:* This step in the process evaluates the alternative feasible designs with respect

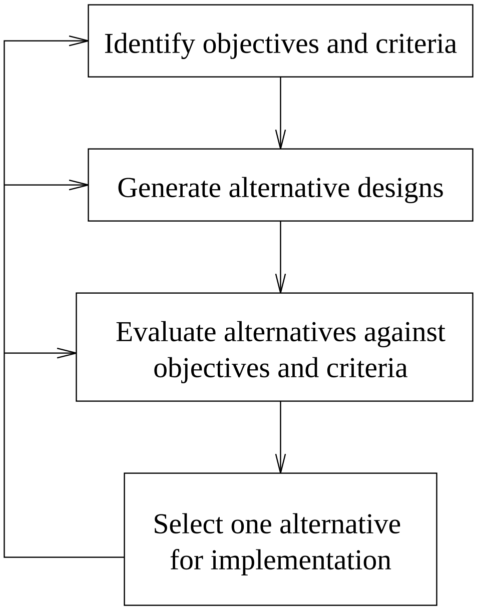

Identify objectives and criteria
Generate alternative designs
Evaluate alternatives against objectives and criteria
Select one alternative for implementation

Figure 4.3: The Standard System Engineering Process.

to various design criteria. A hazard might be controlled by any one of several safeguards: A trade study would determine the relative desirability of each safeguard with respect to effectiveness, cost, weight, size, safety, and any other relevant criteria. For example, substitution of one material for another may reduce the risk of fire or explosion, but may also reduce reliability or efficiency. Each alternative design may have its own set of safety constraints (derived from the system hazards) as well as other performance goals and constraints that need to be assessed. Although, ideally, decisions should be based upon mathematical analysis, quantification of many of the key factors is often difficult, if not impossible, and subjective judgment often has to be used.

- *System architecture development and analysis:* In this step, the system engineers break down the system into a set of subsystems, together with the functions and constraints—including safety constraints—imposed upon the individual subsystem designs, the major system interfaces, and the subsystem interface topology. These aspects are analyzed with respect to desired system performance characteristics and constraints (again including safety constraints) and the process is iterated until an acceptable system design results. The preliminary design at the end of this process must be described in sufficient detail that subsystem implementation can proceed independently.

- *Interface analysis:* The interfaces define the functional boundaries of the system components. From a management standpoint, interfaces must (1) optimize visibility and control and (2) isolate components that can be implemented independently and for which authority and responsibility can be delegated [89]. From an engineering standpoint, interfaces must be designed to separate independent functions and to facilitate the integration, testing, and

> operation of the overall system. Because interfaces tend to be particularly susceptible to design error and are implicated in the majority of accidents, a paramount goal of interface design is simplicity. Simplicity aids in ensuring that the interface can be adequately designed and tested prior to integration and that interface responsibilities can be clearly understood.

Any specific realization of this general systems engineering process depends on the engineering models used for the system components and the desired system qualities. For safety, the models commonly used to understand why and how accidents occur have been based on events, particularly failure events, and the use of reliability engineering techniques to prevent them. Part II of this book proposes an alternative. The most useful accident models will go beyond simple component failure and reliability and will support the systems engineering problem-solving model and its related processes.

Part II

STAMP: An Accident Model Based on Systems Theory

Part II introduces an accident model based on systems theory called STAMP—Systems Theory Accident Modeling and Processes. In this conception of safety, accidents (loss events[1]) occur when external disturbances, component failures, and/or dysfunctional interactions among system components are not adequately controlled, i.e., accidents result from inadequate control or enforcement of safety-related constraints on the development, design, and operation of the system.

Safety then can be viewed as a control problem, and, as suggested by Rasmussen, safety is managed by a control structure embedded in an adaptive socio-technical system. The goal of the safety control structure is to enforce safety-related constraints (1) on system development, including both the development process itself and the resulting system design, and (2) on system operation.

In this framework, understanding why an accident occurred requires determining why the control structure was ineffective. Preventing future accidents requires designing a control structure that will enforce the necessary constraints. Thus the most basic concept in STAMP is not an event, but a constraint.

Systems are viewed, in this approach, as interrelated components that are kept in a state of dynamic equilibrium by feedback loops of information and control. A system is not treated as a static design but as a dynamic process that is continually adapting to achieve its ends and to react to changes in itself and its environment. The original design must not only enforce appropriate constraints on behavior to ensure safe operation, but it must continue to operate safely as changes and adaptations occur over time. Accidents then are viewed as the result of flawed processes involving interactions among people, societal and organizational structures, engineering activities, and physical system components. The process leading up to an accident can be described in terms of an adaptive feedback function that fails to maintain safety as performance changes over time to meet a complex set of goals and values.

Instead of defining safety management in terms of preventing component failure events, it is defined as a continual control task to impose the constraints necessary to limit system behavior to safe changes and adaptations. Accidents can be understood, using STAMP, in terms of why the controls that were in place did not prevent or detect maladaptive changes, that is, by identifying the safety constraints that were violated and determining why the controls were inadequate in enforcing them. For example, understanding the Bhopal accident requires not simply determining why the maintenance personnel did not insert the slip blind (an inadvertent omission or sabotage), but why the controls that had been designed into the system to prevent the release of hazardous chemicals and to mitigate the consequences of such occurrences—including refrigeration units, gauges and other monitoring units, a vent scrubber, water spouts, a flare tower, safety audits, alarms and practice alerts, and emergency procedures and equipment—were not successful.

The three chapters in Part II examine the three basic components of STAMP: constraints, hierarchical levels of control, and process loops. A classification of accident factors is derived from these three components. Then Part III shows how STAMP can be used as a basis for new and more effective accident analysis, hazard analysis and prevention, risk assessment, and safety metrics.

[1] Definitions are always controversial, but a set of definitions for the safety terms as used in this book can be found in Appendix A

Chapter 5

The Central Role of Constraints in System Safety

Constraints on behavior are at the core of STAMP. In systems theory, control is always associated with the imposition of constraints. Instead of viewing accidents as the result of an initiating event in a series of events leading to a loss event, they are considered to result from a lack constraints imposed on the system design and operation, that is, by inadequate enforcement of constraints on behavior at each level (e.g., technical, managerial, regulatory) of a socio-technical system. The safety-related constraints specify those relationships between system variables that constitute the nonhazardous system states. An example of a safety constraint at the physical system level is: *the power must never be on when the access door is open.* The control processes that enforce these constraints limit system behavior to safe changes and adaptations.

Constraints (versus failures) are particularly useful in explaining and preventing accidents where design errors, particularly those related to software and operator behavior, play a part.

5.1 Constraints and Software Design

Why do design constraints play such an important role in the safety of complex systems, particularly software-intensive systems? The computer is so powerful and so useful because it has eliminated many of the physical constraints of electromechanical devices. This is both its blessing and its curse: We no longer have to worry about the physical realization of our designs, but that implies that we no longer have physical laws the limit the complexity of these designs. I call this the *curse of flexibility.* Physical constraints enforce discipline on the design, construction, and modification of our design artifacts. Physical constraints also control the complexity of what we build. With software, the limits of what is *possible* to accomplish are different than the limits of what can be accomplished *successfully* and *safely*—the limiting factors change from the structural integrity and physical constraints of our materials to limits on our intellectual capabilities. It is possible and even quite easy to build software (i.e., system designs) that we cannot understand in terms of being able to determine how it will behave under all conditions: We can construct software (and often do) that goes beyond human intellectual limits. The result has been an increase in system accidents stemming from intellectual unmanageability related to interactively complex and tightly coupled designs that allow potentially unsafe interactions to go undetected during development.

The solution to this problem is for engineers to enforce the same discipline on the software parts of the system design that nature imposes on the physical parts. Safety, like any quality, must be built into the system design. When software acts as a controller in complex systems, it represents or *is* the system design—it embodies or enforces the system safety constraints by controlling the components and their interactions. Control software, then, contributes to an accident by not enforcing the appropriate constraints on behavior or by commanding behavior that violates the constraints. In the batch reactor example of Section 3.2, a system safety constraint related to the software is that water must be flowing into the reflux condenser whenever the flow of catalyst to the reactor is initiated. This system behavioral constraint translates to a constraint on software behavior (a software requirement) that the software must always open the water valve before the catalyst valve.

STAMP provides a much better description of how software affects accidents than a failure model and partially explains why most software-related accidents stem from requirements errors (misunderstanding about the required software behavior) rather than coding errors. The primary safety problem in computer-controlled systems is not software "failure" but the lack of appropriate constraints on software behavior, and the solution is to identify the required constraints and enforce them in the software and overall system design. The job of the system engineer or system safety engineer is to identify the constraints necessary to ensure safe system behavior and effectively communicate these behavioral constraints to the software engineers who, in turn, must enforce them in their software. This process is detailed further in Chapter 9.

5.2 Constraints and Human–Automation Interface Design

The relaxation of previous physical constraints also impacts human supervision and control of automated systems and the design of interfaces between operators and controlled processes [21]. Cook argues that when controls were primarily mechanical and were operated by people located close to the operating process, proximity allowed sensory perception of the status of the process via direct physical feedback such as vibration, sound, and temperature. Displays were directly linked to the process and thus were essentially a physical extension of it. For example, the flicker of a gauge needle in the cab of a train indicated (1) the engine valves were opening and closing in response to slight pressure fluctuations, (2) the gauge was connected to the engine, (3) the pointing indicator was free, etc. In this way, the displays provided a rich source of information about the controlled process and the state of the displays themselves.

The introduction of electromechanical controls allowed operators to control the process from a greater distance (both physical and conceptual) than possible with pure mechanically linked controls. That distance, however, meant that operators lost a lot of direct information about the process—they could no longer sense the process state directly and the control and display surfaces no longer provided as rich a source of information about the process (or the state of the controls themselves). The system designers had to synthesize and provide an image of the process state to the operators. An important new source of design errors was the need for the designers to determine beforehand what information the operator would need under all conditions to safely control the process. If the designers had not anticipated a particular situation could occur and provided for it in the original system design, they might also not anticipate the need of the operators for information about it during operations.

Designers also had to provide feedback on the actions of the operators and on any failures that might have occurred. The controls could now be operated without the desired effect on the process, and the operators might not know about it. Accidents started to occur due to incorrect feedback. For example, major accidents (including Three Mile Island) have involved the operators commanding a valve to open and receiving feedback that the valve had opened as a result, when in reality it had not. In this case and others, the valves were wired to provide feedback that power had been applied to the valve, but not that it had actually opened. Not only could the design of the feedback about failures be misleading, but the return links were also subject to failure themselves.

Thus, electromechanical controls relaxed constraints on the system design allowing greater functionality. At the same time, they created new possibilities for designer and operator error that had not existed or were much less likely in mechanically controlled systems. The later introduction of computer and digital controls afforded additional advantages and removed even more constraints on the control system design—and introduced more possibility for error.

It is the freedom from constraints that makes the design of such systems so difficult. The physical constraints shaped system design in ways that efficiently transmitted valuable physical component and process information and supported the operators' cognitive processes. Proximity provided rich sources of feedback that involved almost all of the senses, enabling early detection of potential problems. We are finding it hard to capture and provide these same qualities in new systems that use computer controls and displays.

In summary, accidents result from a lack of appropriate constraints on system design. The role of the system engineer or system safety engineer is to identify the design constraints necessary to maintain safety and to ensure that the system design, including the social and organizational aspects of the system and not just the physical ones, enforce them.

Chapter 6

Hierarchical Safety Control Levels

In systems theory (see Section 4.3), systems are viewed as hierarchical structures where each level imposes constraints on the activity of the level beneath it—that is, constraints or lack of constraints at a higher level allow or control lower-level behavior. In a systems theory model of accidents, instead of decomposing systems (and accident causation) into structural components and a flow of events, systems and accidents are described in terms of a hierarchy of control based on adaptive feedback mechanisms. This approach allows adaptation to play a central role in the understanding and prevention of accidents. The first step is to create a hierarchical control model.

6.1 Hierarchical Control Models

Socio-technical systems can be modeled as a hierarchy of levels of organization with control processes operating at the interfaces between levels to control processes at the lower levels. At each level, inadequate control may result from missing constraints, inadequately communicated constraints, constraints that were not enforced correctly at a lower level, or inadequately communicated or processed feedback about constraint enforcement.

The idea of modeling socio-technical systems using system theory concepts is not a new one. For example, Jay Forrester in the 1960s created *system dynamics* using such an approach. System dynamics over the years has been applied to a wide variety of open system problems. The Rasmussen and Svedung model (see Figure 3.3) uses some system theory concepts but, reverts to an event model at each hierarchical level.

STAMP builds on the ideas used in the upper levels of the Ramussen/Svedung model and adds a system development control structure. The control-theoretic approach is continued down through and including the technical system. System dynamics is used to model the reasons for system change or adaptation over time, as shown in Part III.

Figure 6.1 shows a typical socio-technical control model in STAMP. Each system, of course, must be modeled to include its specific features. Examples are provided later in this chapter and in Chapter 8. The generic model shown in Figure 6.1 has two basic hierarchical control structures—one for system development (on the left) and one for system operation (on the right)—with interactions between them. An aircraft manufacturer, for example, might only have system development under its immediate control, but safety involves both development and operational use of the aircraft, and neither can be accomplished successfully in isolation: Safety during operation depends partly on

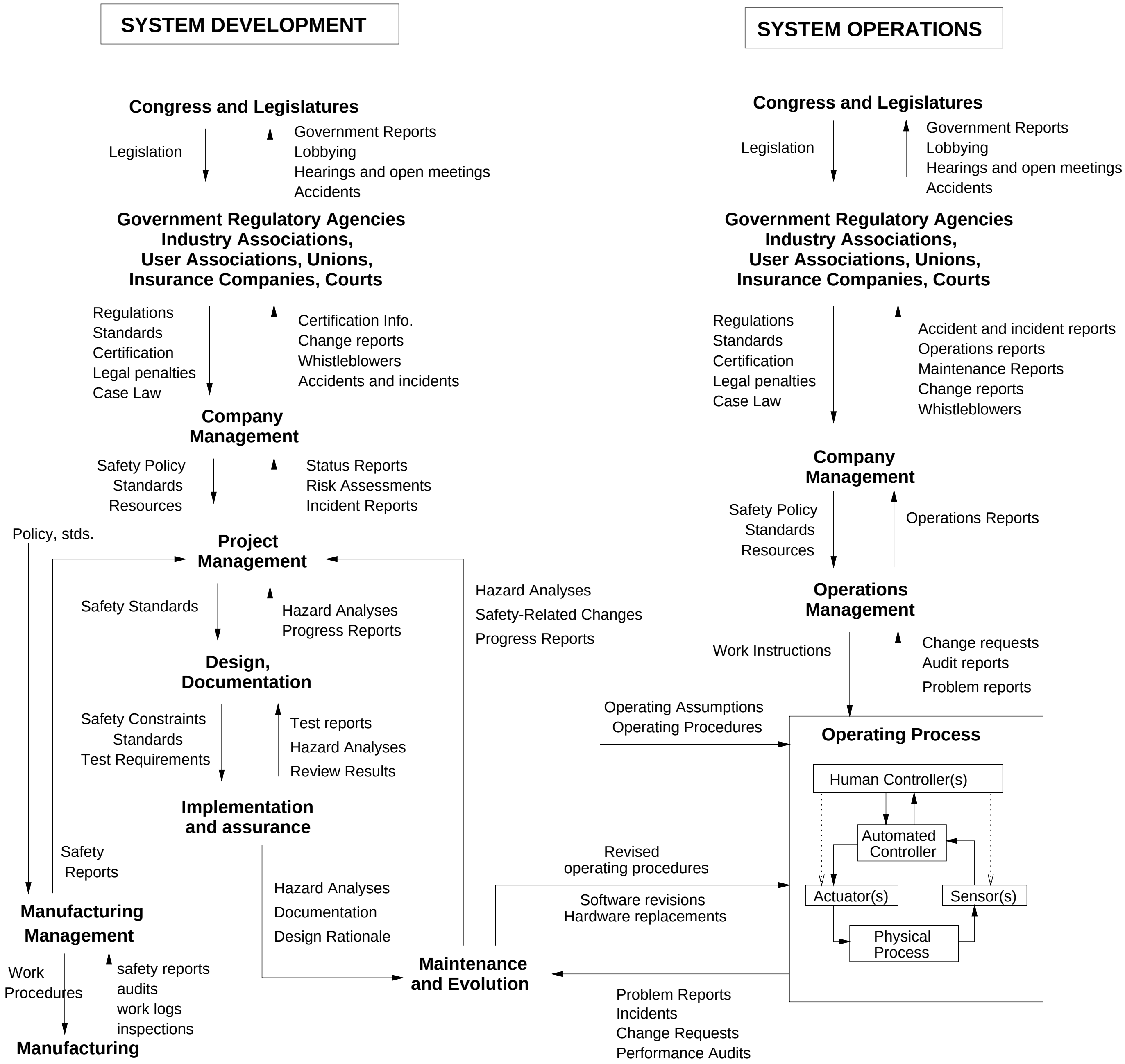


Figure 6.1: General Form of a Model of Socio-Technical Control

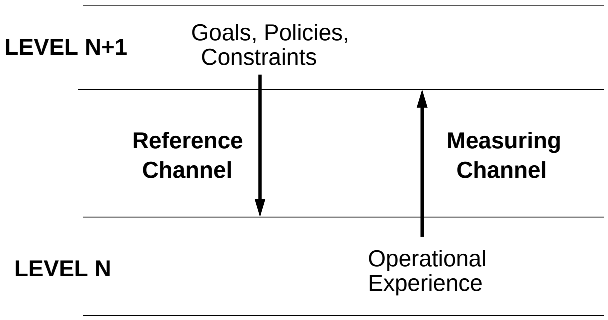


Figure 6.2: Communication Channels Between Control Levels.

the original design and development and partly on effective control over operations. Manufacturers must communicate to their customers the assumptions about the operational environment upon which the safety analysis was based, as well as information about safe operating procedures. The operational environment, in turn, provides feedback to the manufacturer about the performance of the system over its lifetime.

Between the hierarchical levels of each control structure, effective communication channels are needed, both a downward *reference channel* providing the information necessary to impose constraints on the level below and an upward *measuring channel* to provide feedback about how effectively the constraints are being satisfied (Figure 6.2). Feedback is critical in any open system in order to provide adaptive control. As noted above, at each level of the control structure, inadequate control may result from missing constraints, inadequately communicated constraints, or from constraints that are not enforced correctly at a lower level.

Government, general industry groups, and the court system are the top two levels of each of the generic control structures shown in Figure 6.1. The government control structure in place to control development may differ from that controlling operations—a different group at the FAA, for example, is responsible for issuing aircraft type certifications than that responsible for supervising airline operations. The appropriate constraints in each control structure and at each level will vary but in general may include technical design and process constraints, management constraints, manufacturing constraints, and operational constraints.

At the highest level in both the system development and system operation hierarchies are Congress and state legislatures.[1] Congress controls safety by passing laws and by establishing and funding government regulatory structures. Feedback as to the success of these controls or the need for additional ones comes in the form of government reports, congressional hearings and testimony, lobbying by various interest groups, and, of course, accidents.

The next level contains government regulatory agencies, industry associations, user associations, insurance companies, and the court system. Unions may play a role in ensuring safe system operations (such as the air traffic controllers union) or worker safety in manufacturing. The legal system tends to be used when there is no regulatory authority and the public has no other means to encourage a desired level of concern for safety in company management. The constraints gener-

[1] Obvious changes are required in the model for countries other than the U.S. The U.S. is used in the example here because of the author's familiarity with it.

ated at this level and passed down to the companies are usually in the form of policy, regulations, certification, standards (by trade or user associations), or threat of litigation. Where there is a union, safety-related constraints on operations or manufacturing may result from union demands and collective bargaining.

Company management takes the standards, regulations, and other general controls on its behavior and translates them into specific policy and standards for the company. Many companies have a general safety policy (it is required by law in Great Britain) as well as more detailed standards documents. Feedback may come in the form of status reports, risk assessments, and incident reports.

In the development control structure (shown on the left of Figure 6.1), the company policies and standards are tailored and usually augmented by each engineering project to fit the needs of the particular project. The higher-level control process may provide only general goals and constraints and each level may then add many details to operationalize the general goals and constraints given the immediate conditions and local goals. For example, while government and/or company standards may require a hazard analysis be performed, the system designers and documenters (including those designing the operational procedures and writing user manuals) may have control over the actual hazard analysis process used to identify specific safety constraints on the design and operation of the system. The design constraints identified as necessary to control system hazards are passed to the implementers and assurers of the individual system components along with standards and other requirements. Success is determined through test reports, reviews, and various additional hazard analyses. At the end of the development process, the results of the hazard analyses as well as documentation of the safety-related design features and design rationale should be passed on to the maintenance group to be used in the system evolution and sustainment process.

A similar process involving layers of control is found in the system operation control structure. In addition, there will be (or at least should be) interactions between the two structures. For example, the safety design constraints used during development should form the basis for operating procedures and for performance and process auditing.

Rasmussen notes that control at each level may be enforced in a very prescriptive command and control structure (see the example in the next section) or it may be loosely implemented as performance objectives with many degrees of freedom in how the objectives are met. Recent trends from management by *oversight* to management by *insight* reflect differing levels of feedback control that are exerted over the lower levels and a change from prescriptive management control to management by objectives, where the objectives are interpreted and satisfied according to the local context. In a Titan/Centaur/Milstar loss [86] and in the NASA Mars '98 accidents (Mars Climate Orbiter [109] and Mars Polar Lander [50, 130]), the accident reports note that a poor transition from oversight to insight was a factor in the losses. Attempts to delegate decisions and to manage by objectives requires an explicit formulation of the value criteria to be used and an effective means for communicating the values down through society and organizations. And the impact of specific decisions at each level on the objectives and values passed down need to be adequately and formally evaluated. Feedback is required to measure how successfully the functions were performed.

As in any control loop, time lags may affect the flow of control actions and feedback and may impact the efficiency of the control loop. For example, standards can take years to develop or change—a time scale that may keep them behind current technology and practice. At the physical level, new technology may be introduced in different parts of the system at different rates (*asynchronous evolution* of the control structure). In the accidental shootdown of two U.S. Army

Black Hawk helicopters by two U.S. Air Force F-15s in the No-Fly-Zone over northern Iraq in 1994, for example, the fighter jet aircraft and the helicopters were inhibited in communicating by radio because the F-15's had newer jam-resistant radios that could not communicate with the older technology Army helicopter radios. Hazard analysis needs to include the influence of these time lags.

In general, a common way to deal with time lags is to delegate responsibility to lower levels that are not subject to as great a delay in obtaining information or feedback from the measuring channels. In periods of quickly changing technology, time lags may make it necessary for the lower levels to augment the control processes passed down from above or to modify them to fit the current situation. Time lags at the lowest levels, as in the Black Hawk shootdown example, may require the use of feedforward control to overcome lack of feedback or instituting controls on behavior: Communication between the F-15s and the Black Hawks would have been possible if the F-15 pilots had been told to use an older radio technology available to them (as they were commanded to do for other types of friendly aircraft).

Note that the operation of the socio-technical safety control structure at all levels is facing the stresses noted in Section 1, such as rapidly changing technology, competitive and time-to-market pressures, and changing public and regulatory views of responsibility for safety. These pressures can lead to a need for new procedures or for controls to ensure that required safety constraints are not ignored.

Aside from a few isolated proposals such as MORT, Johnson's attempt to incorporate management factors into a fault tree format [49], accident models and hazard analysis techniques have omitted all the levels of this socio-technical safety control structure except the lowest, technical level. Effective accident analysis and prevention, however, requires going beyond the current approaches.

The next section contains an example of a control structure designed to prevent a hazard, in this case the friendly fire accident mentioned above. The analysis, using STAMP, of why the accident occurred is completed in the next chapter. This example was chosen because the controversy and multiple viewpoints and books about the shootdown provide the information necessary to recreate most of the control structure. Accident reports do not often include that information unless some unusual controversy surrounds the investigation or the accident itself or the report is exceptionally good. Because of the nature of this particular accident, the investigation and analysis was primarily focused on operations. Examples of safety control structures for system development and their relationship to accidents can be found in Chapter 8.

6.2 An Example Hierarchical Safety Control Structure for Friendly Fire

After the Persian Gulf War, Operation Provide Comfort (OPC) was created as a multinational humanitarian effort to relieve the suffering of hundreds of thousands of Kurdish refugees who fled into the hills of northern Iraq during the war. The goal of the military efforts was to provide a safe haven for the resettlement of the refugees and to ensure the security of relief workers assisting them. The formal mission statement for OPC read: "To deter Iraqi behavior that may upset peace and order in northern Iraq."

In addition to operations on the ground, a major component of OPC's mission was to occupy

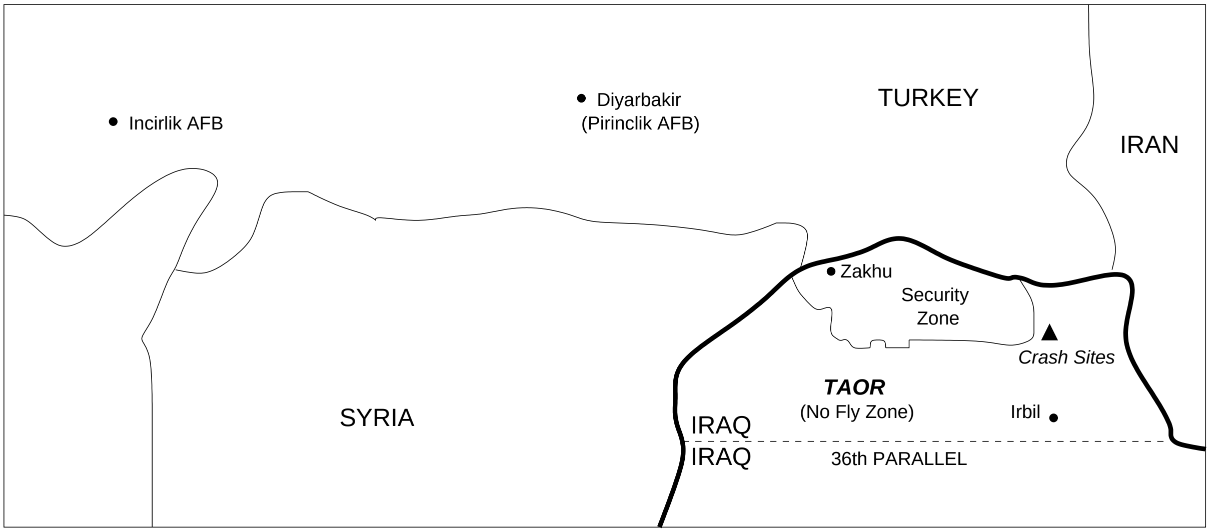


Figure 6.3: The No-Fly-Zone and Relevant Surrounding Locations

the airspace over northern Iraq. To accomplish this task, a no-fly-zone (also called the TAOR or Tactical Area of Responsibility) was established that included all airspace within Iraq north of the 36th parallel (see Figure 6.3). Air operations were led by the Air Force to prohibit Iraqi aircraft from entering the No-Fly-Zone while ground operations were organized by the Army to provide humanitarian assistance to the Kurds and other ethnic groups in the area.

U.S., Turkish, British, and French fighter and support aircraft patrolled the No-Fly-Zone daily to prevent Iraqi warplanes from threatening the relief efforts. The mission of the Army helicopters was to support the ground efforts; the Army used them primarily for troop movement, resupply, and medical evacuation.

On April 15, 1994, after nearly three years of daily operations over the TAOR (Tactical Area of Responsibility), two U.S. Air Force F-15's patrolling the area shot down two U.S. Army Black Hawk helicopters, mistaking them for Iraqi Hind helicopters. The Black Hawks were carrying 26 people including 15 U.S. citizens and 11 others, among them British, French, and Turkish military officers as well as Kurdish citizens. All were killed in one of the worst air-to-air friendly fire accidents involving U.S. aircraft in military history.

All the aircraft involved were flying in clear weather with excellent visibility, an AWACS (Airborne Warning and Control System) aircraft was providing surveillance and control for the aircraft in the area, and all the aircraft were equipped with electronic identification and communication equipment (apparently working properly) and flown by decorated and highly experienced pilots.

The hazard being controlled was mistaking a "friendly" (coalition) aircraft for a threat and shooting at it. This hazard, informally called *friendly fire*, was well known and a control structure was established to prevent it. Appropriate constraints were established and enforced at each level, from the Joint Chiefs of Staff down to the aircraft themselves. Understanding why this accident occurred requires understanding why the control structure in place was ineffective in preventing the loss. Preventing future accidents involving the same control flaws requires making appropriate changes to the control structure, including establishing monitoring and feedback loops to detect when the controls are becoming ineffective and the system is migrating toward an accident (moving

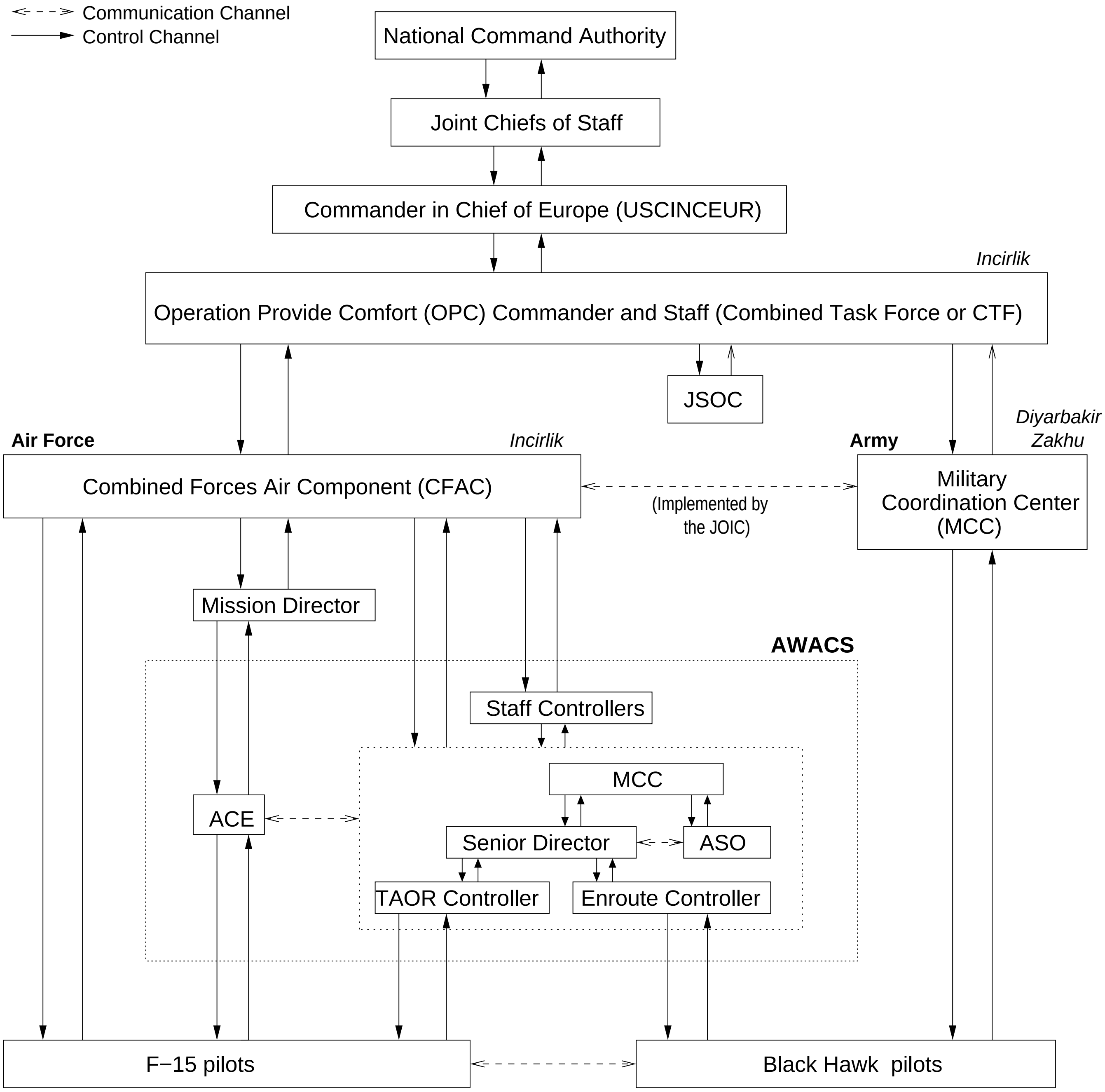

Communication Channel
Control Channel
National Command Authority
Joint Chiefs of Staff
Commander in Chief of Europe (USCINCEUR)
Incirlik
Operation Provide Comfort (OPC) Commander and Staff (Combined Task Force or CTF)
JSOC
Air Force
Incirlik
Army
Diyarbakir
Zakhu
Combined Forces Air Component (CFAC)
(Implemented by the JOIC)
Military Coordination Center (MCC)
Mission Director
AWACS
Staff Controllers
MCC
ACE
Senior Director
ASO
TAOR Controller
Enroute Controller
F−15 pilots
Black Hawk pilots

Figure 6.4: Control Structure in the Iraqi No-Fly Zone.

toward a state of increased risk). The more comprehensive the model and factors identified, the larger the class of accidents that can be prevented. This section describes the safety control structure for this accident while Chapter 7 contains an analysis of the accident using STAMP.

For this example analysis, information about the accident and the control structure was obtained from the original accident report [4], a GAO (Government Accounting Office) report on the accident investigation process and results [113], and two books on the shootdown—one a Ph.D. dissertation by Scott Snook [108] and one by the mother (Joan Piper) of one of the victims [90]. Because of the extensive existing analysis, much of the control structure (shown in Figure 6.4) can be reconstructed from these sources.

National Command Authority and Commander-in-Chief Europe

When the National Command Authority (the President and Secretary of Defense) directed the military to conduct Operation Provide Comfort (OPC), the U.S. Commander in Chief Europe (USCINCEUR) directed the creation of Combined Task Force (CTF) Provide Comfort.

A series of orders and plans established the general command and control structure of the CTF. These orders and plans also transmitted sufficient authority and guidance to subordinate component commands and operational units so that they could then develop the local procedures that were necessary to bridge the gap between general mission orders and specific subunit operations.

At the top of the control structure, the National Command Authority (the President and Secretary of Defense, who operate through the Joint Chiefs of Staff) provided guidelines for establishing Rules of Engagement (ROE). ROE govern the actions allowed by U.S. military forces to protect themselves and other personnel and property against attack or hostile incursion and specify a strict sequence of procedures to be followed prior to any coalition aircraft firing its weapons. They are based on legal, political, and military considerations and are intended to provide for adequate self defense to ensure that military activities are consistent with current national objectives and that appropriate controls are placed on combat activities. Commanders establish ROE for their areas of responsibility that are consistent with the Joint Chiefs of Staff guidelines, modifying them for special operations and for changing conditions.

Because the ROE dictate how hostile aircraft or military threats are treated, they play an important role in any friendly fire accidents. The ROE in force for OPC were the peacetime ROE for the United States European Command with OPC modifications approved by the National Command Authority. These conservative ROE required a strict sequence of procedures to be followed prior to any coalition aircraft firing its weapons. The less aggressive peacetime rules of engagement were used even though the area had been designated a combat zone because of the number of countries involved in the joint task force. The goal was to slow down any military confrontation, thus preventing the type of friendly fire accidents that had been common during Operation Desert Storm. Thus the ROE were an important control in force to prevent the type of accident involved in the shootdown of the Black Hawk helicopters and understanding the reasons for the accident requires understanding why the ROE did not provide effective controls against friendly fire.

Safety Constraints Related to the Accident:

1. The NCA and UNCINCEUR must establish a command and control structure that provides the ability to prevent friendly fire accidents.

2. The guidelines for ROE generated by the Joint Chiefs of Staff (with tailoring to suit specific operational conditions) must be capable of preventing friendly fire accidents in all types of situations.

3. The European Commander-in-Chief must review and monitor operational plans generated by the Combined Task Force, ensure they are updated as the mission changes, and provide the personnel required to carry out the plans.

Controls: The controls in place included the ROE guidelines, the operational orders, and review procedures for the controls generated as a result (e.g., the actual ROE and Operational Plans) at the control levels below.

Combined Task Force (CTF)

The components of the Combined Task Force (CTF) organization relevant to the accident (and to preventing friendly fire) were a Combined Task Force staff, a Combined Forces Air Component (CFAC), and an Army Military Coordination Center. The Air Force fighter aircraft were co-located with CTF Headquarters and CFAC at Incirlik Air Base in Turkey while the U.S. Army helicopters were located with the Army headquarters at Diyarbakir, also in Turkey (see Figure 6.3).

The Combined Task Force had three components under it (Figure 6.4):

1. The Military Coordination Center (MCC) monitored conditions in the security zone and had operational control of Eagle Flight helicopters (the Black Hawks), which provided general aviation support to the MCC and the CTF.

2. The Joint Special Operations Component (JSOC) was assigned primary responsibility to conduct search and rescue operations should any coalition aircraft go down inside Iraq.

3. The Combined Forces Air Component (CFAC) was tasked with exercising tactical control of all OPC aircraft operating in the Tactical Area of Responsibility (TAOR) and operational control over Air Force aircraft.[2] The CFAC commander exercised daily control of the OPC flight mission through a Director of Operations (CFAC/DO), as well as a ground-based Mission Director at the Combined Task Force (CTF) headquarters in Incirlik and an Airborne Command Element aboard the AWACS.

Operational orders were generated at the European Command level of authority that defined the initial command and control structure and directed the CTF commanders to develop an operations plan to govern OPC. In response, the CTF commander created an operations plan in July 1991 delineating the command relationships and organizational responsibilities within the CTF. In September 1991, the U.S. Commander in Chief, Europe, modified the original organizational structure in response to the evolving mission in northern Iraq, directing an increase in the size of the Air Force and the withdrawal of a significant portion of the ground forces. The CTF was ordered to provide a supporting plan to implement the changes necessary in their CTF operations

[2] Tactical control involves a fairly limited scope of authority, i.e., the detailed and usually local direction and control of movement and maneuvers necessary to accomplish the assigned mission. Operational control, on the other hand, involves a broader authority to command subordinate forces, assign tasks, designate objectives, and give the authoritative direction necessary to accomplish the mission.

plan. The Accident Investigation Board found that although an effort was begun in 1991 to revise the operations plan, no evidence could be found in 1994 to indicate that the plan was actually updated to reflect the change in command and control relationships and responsibilities. The critical element of the plan with respect to the shootdown was that the change in mission led to the departure of an individual key to the communication between the Air Force and Army, without his duties being assigned to someone else.

In summary, the safety constraints at the highest levels of the socio-technical operations control structure were:

Safety Constraints Related to the Accident:

1. Rules of engagement and operational orders and plans must be established at the Command level that prevent friendly fire accidents. The plans must include allocating responsibility and establishing and monitoring communication channels to allow for coordination of flights into the theater of action.

2. Compliance with the ROE and operational orders and plans must be monitored. Alterations must be made in response to changing conditions and changing mission.

Controls: The controls included the ROE and operational plans plus feedback mechanisms on their effectiveness and application.

CFAC and MCC

The two parts of the Combined Task Force involved in the accident were the Army Military Coordination Center (MCC) and the Air Force Combined Forces Air Component (CFAC).

The shootdown obviously involved a communication failure—the F-15 pilots did not know the U.S. Army Black Hawks were in the area or that they were targeting friendly aircraft. Problems in communication between the three services (Air Force, Army, and Navy) are legendary. Procedures had been established to attempt to eliminate these problems in Operation Provide Comfort.

The Military Coordination Center (MCC) coordinated land and U.S. helicopter missions that supported the Kurdish people. In addition to providing humanitarian relief and protection to the Kurds, another important function of the Army detachment was to establish an ongoing American presence in the Kurdish towns and villages by showing the U.S. flag. This U.S. Army function was supported by a helicopter detachment called Eagle Flight.

All CTF components, with the exception of the Army Military Coordination Center lived and operated out of Incirlik Air Base in Turkey. The MCC operated out of two locations. A forward headquarters was located in the small village of Zakhu (see Figure 6.3), just inside Iraq. Approximately twenty people worked in Zakhu, including operations, communications, and security personnel, medics, translators, and coalition chiefs. Zakhu operations were supported by a small administrative contingent working out of Pirinclik Air Base in Diyarbakir, Turkey. Pirinclik is also where the Eagle Flight Platoon of UH-60 Black Hawk helicopters was located. Eagle Flight helicopters made numerous (usually daily) trips to Zakhu to support MCC operations.

The Combined Forces Air Component (CFAC) Commander was responsible for coordinating the employment of all air operations to accomplish the OPC mission. He was delegated operational control of the Airborne Warning and Control System (AWACS), USAF airlift, and fighter forces.

He had tactical control of the U.S. Army, U.S. Navy, Turkish, French, and British fixed wing and helicopter aircraft This splintering of control, along with communication problems, were major contributors to the accident.

In a complex coordination problem of this sort, communication is critical. Communications were implemented through the Joint Operations and Intelligence Center (JOIC). The JOIC received, delivered, and transmitted communications up, down, and across the CTF control structure. No Army Liaison Officer was assigned to the JOIC, but one was available on request to provide liaison between the MCC helicopter detachment and the CTF staff.

To prevent friendly fire accidents, pilots need to know exactly what friendly aircraft are flying in the No-Fly-Zone at all times as well as know and follow the ROE and other procedures for preventing such accidents. The higher levels of control delegated to the CTF level and below the authority and guidance to develop local procedures,[3] which included:

- Airspace Control Order: The ACO contains the authoritative guidance for all local air operations in OPC. It covers such things as standard altitudes and routes, air refueling procedures, recovery procedures, airspace deconfliction responsibilities, and jettison procedures. The deconfliction procedures were a way to prevent interactions between aircraft that might result in accidents. For the Iraqi TAOR, fighter aircraft, which usually operated at high altitudes, were to stay above 10,000 feet above ground level while helicopters, which normally conducted low-altitude operations, were to stay below 400 feet. All flight crews were responsible for reviewing and complying with the information contained in the ACO. The CFAC Director of Operations was responsible for publishing guidance, including the Airspace Control Order, for conduct of OPC missions.

- Aircrew Read Files (ARFs): The Aircraft Read Files supplement the ACOs and are also required reading by all flight crews. They contain the classified rules of engagement (ROE), changes to the ACO, and recent amplification of how local commanders want air missions executed.

- Air Tasking Orders (ATOs): While the ACO and ARFs contain general information that applies to all aircraft in OPC, specific mission guidance was published in the daily ATOs. They contained the daily flight schedule, radio frequencies to be used, IFF codes (used to identify an aircraft as friend or foe), and other late-breaking information necessary to fly on any given day. All aircraft are required to have a hard copy of the current ATO with Special Instructions (SPINS) on board before flying. Each morning around 1130, the mission planning cell (or Frag shop) publishes the ATO for the following day, and copies are distributed to all units by late afternoon.

- Battle Staff Directives (BSDs): Any late scheduling changes that do not make it onto the ATO are published in last minute Battle Staff Directives, which are distributed separately and attached to all ATOs prior to any missions flying the next morning.

- Daily Flow Sheets: Military pilots fly with a small clipboard attached to their knees. These kneeboards contain boiled-down reference information essential to have handy while flying a mission, including the daily flow sheet and radio frequencies. The flow sheets are graphical

[3] The term *procedures* as used in the military denote standard and detailed courses of action that describe how to perform a task.

depictions of the chronological flow of aircraft scheduled into the no-fly-zone for that day. Critical information is taken from the ATO, translated into timelines, and reduced on a copier to provide pilots with a handy inflight reference.

- Local Operating Procedures and Instructions, Standard Operating Procedures, Checklists, etc.

In addition to written material, real-time guidance is provided to pilots after taking off via radio through an unbroken command chain that runs from the OPC Commanding General, through the CFAC, through the Mission Director, through an Airborne Command Element (ACE) on board the AWACS, and ultimately to pilots.

The CFAC commander of operations was responsible for ensuring aircrews were informed of all unique aspects of the OPC mission, including the ROE, upon their arrival. He was also responsible for publishing the Aircrew Read File (ARF), the Airspace Control Order (ACO), the daily Air Tasking Order, and mission-related special instructions (SPINS).

Safety Constraints Related to the Accident:

1. Coordination and communication among all flights into the TAOR must be established. Procedures must be established for determining who should be and is in the TAOR at all times.

2. Procedures must be instituted and monitored to ensure that all aircraft in the TAOR are tracked and fighters are aware of the location of all friendly aircraft in the TAOR.

3. The ROE must be understood and followed by those at lower levels.

4. All aircraft must be able to communicate effectively in the TAOR.

Controls: The controls in place included the ACO, ARFs, flowsheets, intelligence and other briefings, training (on the ROE, on aircraft identification, etc.), AWACS procedures for identifying and tracking aircraft, established radio frequencies and radar signals for the No-Fly-Zone, a chain of command (OPC Commander to Mission Director to ACE to pilots), disciplinary actions for those not following the written rules, and a group (the JOIE) responsible for ensuring effective communication occurred.

Mission Director and Airborne Command Element:

The Airborne Command Element (ACE) flies in the AWACS and is the commander's representative in the air, armed with up-to-the-minute situational information to make time-critical decisions. The ACE monitors all air operations and is in direct contact with the Mission Director located in the ground command post. He must also interact with the AWACS crew to identify reported unidentified aircraft.

The ground-based Mission Director maintains constant communication links with both the ACE up in the AWACS and with the CFAC commander on the ground. The Mission Director must inform the OPC commander immediately if anything happens over the No-Fly-Zone that might require a decision by the commander or his approval. Should the ACE run into any situation that would involve committing U.S. or coalition forces, the Mission Director will communicate with him to provide command guidance. The Mission Director is also responsible for making weather-related decisions, implementing safety procedures, scheduling aircraft, and ensuring that the ATO is executed correctly.

The ROE in place at the time of the shootdown stated that aircrews experiencing unusual circumstances were to pass details to the ACE or AWACS, who would provide guidance on the appropriate response [113]. Exceptions were possible, of course, in cases of imminent threat. Aircrews were directed to first contact the ACE and, if that individual was unavailable, to then contact AWACS. The six unusual circumstances/occurrences to be reported, as defined in the ROE, included "any intercept run on an unidentified aircraft." As stated, the ROE was specifically designed to slow down a potential engagement to allow time for those in the chain of command to check things out.

Although the written guidance was clear, there was controversy with respect to how it was or should have been implemented and who had decision-making authority. Conflicting testimony during the investigation of the shootdown about responsibility either reflect after-the-fact attempts to justify actions or they reflect real confusion on the part of everyone, including those in charge, as to where the responsibility lay.

Safety Constraints Related to the Accident:

1. The ACE and MD must follow procedures specified and implied by the ROE.
2. The ACE must ensure that pilots follow the ROE.
3. The ACE must interact with the AWACS crew to identify reported unidentified aircraft.

Controls: Controls to enforce the safety constraints included: the ROE to slow down engagements and a chain of command to prevent individual error or erratic behavior; the ACE up in the AWACS to get up-to-the-minute information about the state of the TAOR airspace; and the Mission director on the ground to provide a chain of command from the pilots to the CFAC commander for real-time decision making.

AWACS Controllers

The AWACS (Airborne Warning and Control Systems) acts as an air traffic control tower in the sky. The AWACS OPC mission was to:

1. Control aircraft enroute to and from the No-Fly-Zone;
2. Coordinate air refueling (for the fighter aircraft and the AWACS itself);
3. Provide airborne threat warning and control for all OPC aircraft operating inside the No-Fly-Zone.
4. Provide surveillance, detection, and identification of all unknown aircraft.

An AWACS is a modified Boeing 707, with a saucer-shaped radar dome on the top, equipped inside with powerful radars and radio equipment that scan the sky for aircraft. A computer takes raw data from the radar dome, processes it, and ultimately displays tactical information on 14 color consoles arranged in rows of three throughout the rear of the aircraft. AWACS have the capability to track approximately 1000 enemy aircraft at once while directing 100 friendly ones [90].

The AWACS carries a flight crew (pilot, copilot, navigator, and flight engineer) responsible for safe ground and flight operation of the AWACS aircraft and a mission crew that has overall responsibility for the AWACS command, control, surveillance, communications, and sensor systems.

The mission crew of approximately 19 people are under the direction of a Mission Crew Commander (MCC). The Mission Crew Commander has overall responsibility for the AWACS mission and the management, supervision, and training of the mission crew. The mission crew were divided into three sections:

1. **Technicians**: The technicians are responsible for operating, monitoring, and maintaining the physical equipment on the aircraft.

2. **Surveillance**: The surveillance section is responsible for the detection, tracking, identification, height measurement, display, and recording of surveillance data. As unknown targets appear on the radarscopes, surveillance technicians follow a detailed procedure to identify the tracks. They are responsible for handling unidentified and non-OPC aircraft detected by the AWACS electronic systems. The section is supervised by the Air Surveillance Officer and the work is carried out by an Advanced Air Surveillance Technician and three Air Surveillance Technicians.

3. **Weapons**: The weapons controllers are supervised by the Senior Director (SD). This section is responsible for the control of all assigned aircraft and weapons systems in the TAOR (tactical area of operations). The SD and three weapons directors are together responsible for locating, identifying, tracking, and controlling all friendly aircraft flying in support of OPC. Each weapons director was assigned responsibility for a specific task:

   - The Enroute Controller controlled the flow of OPC aircraft to and from the TAOR. This person also conducted radio and IFF checks on friendly aircraft outside the TAOR.
   - The TAOR Controller provided threat warning and tactical control for all OPC aircraft within the TAOR.
   - The Tanker Controller coordinated all air refueling operations (and played no part in the accident).

To facilitate communication and coordination, the SD's console was physically located in the "pit" right between the Mission Crew Commander and the ACE (Airborne Command Element). Through internal radio nets, the SD synchronized the work of the weapons section with that of the surveillance section. He also monitored and coordinated the actions of his weapons directors to meet the demands of both the ACE and Mission Crew Commander.

Because those who had designed the control structure recognized the potential for some distance to develop between the training of the AWACS crew members and the continually evolving practice in the No-Fly-Zone, they had instituted a control by creating *staff* or instructor personnel permanently stationed in Turkey. Their job was to help provide continuity for U.S. AWACS crews who rotated through OPC on temporary duty status, usually for 30 day rotations. This *shadow crew* flew with each new AWACS crew on their first mission in the TAOR to alert them as to how things were *really* done in OPC. Their job was to answer any questions the new crew might have about local procedures, recent occurrences, or changes in policy or interpretation that had come about since the last time they had been in the theater. Because the accident occurred on the first day for a new AWACS crew, instructor or staff personnel were also on board.

In addition to all these people, a Turkish controller flew on all OPC missions to help the crew interface with local air traffic control systems.

The AWACS typically takes off from Incirlik AB approximately 2 hours before the first air-refueling and fighter aircraft. Once the AWACS is airborne, the AWACS's systems are brought on-line and a Joint Tactical Information Distribution System (JTIDS[4]) link is established with a Turkish Sector Operations Center (radar site). After the JTIDS link is confirmed, the CFAC airborne command element (ACE) initiates the planned launch sequence for the rest of the force. Normally, within a one hour period, tanker and fighter aircraft take off and proceed to the TAOR in a carefully orchestrated flow. Fighters may not cross the political border into Iraq without AWACS coverage.

Safety Constraints Related to the Accident:

1. The AWACS mission crew must identify and track all aircraft in the TAOR. Friendly aircraft must not be identified as a threat (hostile).

2. The AWACS mission crew must accurately inform fighters about the status of all tracked aircraft when queried.

3. The AWACS mission crew must alert aircraft in the TAOR to any coalition aircraft not appearing on the flowsheet (ATO).

4. The AWACS crew must not fail to warn fighters about any friendly aircraft the fighters are targeting.

5. The JTIDS must provide the ground with an accurate picture of the airspace and its occupants.

Controls: Controls included procedures for identifying and tracking aircraft, training (including simulator missions), briefings, staff controllers, and communication channels. The SD and ASO provided real-time oversight of the crew's activities.

Pilots

Fighter aircraft, flying in formations of two and four aircraft, must always have a clear line of command. In the two-aircraft formation involved in the accident, the lead pilot is completely in charge of the flight and the wingman takes all of his commands from the lead.

The ACO (Airspace Control Order) stipulates that fighter aircraft may not cross the political border into Iraq without AWACS coverage and no aircraft may enter the TAOR until fighters with

[4] The Joint Tactical Information Distribution System acts as a central component of the mission command and control system, providing ground commanders with a real-time downlink of the current air picture from AWACS. This information is then integrated with data from other sources to provide commanders with a more complete picture of the situation.

airborne intercept (AI) radars have searched the TAOR for Iraqi aircraft. Once the AI radar-equipped aircraft have "sanitized" the No-Fly-Zone, they establish an orbit and continue their search for Iraqi aircraft and provide air cover while other aircraft are in the area. When they detect non-OPC aircraft, they are to intercept, identify, and take appropriate action as prescribed by the rules of engagement (ROE) as specified in the ACO.

After the area is sanitized, additional fighters and tankers flow to and from the TAOR throughout the 6–8 hour daily flight schedule. This flying window is randomly selected to avoid predictability.

Communication is important in preventing friendly fire accidents. The U.S. Army Black Hawk helicopters carried a full array of standard avionics, radio, IFF, and radar equipment as well as communication equipment consisting of FM, UHF, and VHF radios. Each day the FM and UHF radios were keyed with classified codes to allow pilots to *talk secure* in encrypted mode. The ACO directed that special frequencies were to be used when flying inside the TAOR.

Due to the line-of-sight limitations of their radios, the high mountainous terrain in northern Iraq, and the fact that helicopters tried to fly at low altitudes to use the terrain to mask them from enemy air defense radars, all Black Hawk flights into the No-Fly-Zone also carried tactical satellite radios (TACSATs). These TACSATS were used to communicate with MCC operations. The helicopters had to land to place the TACSATs in operation; they cannot be operated from inside a moving helicopter.

The F-15's were equipped with avionics, communications and electronic equipment similar to that on the Black Hawks except that the F-15's were equipped with HAVE QUICK II (HQ-II) frequency-hopping radios while the helicopters were not. HQ-II defeated most enemy attempts to jam transmissions by changing frequencies many times per second. Although the F-15 pilots preferred to use the more advanced HAVE QUICK technology, the F-15 radios were capable of communicating in a clear, non-HQ-II mode. The ACO directed that F-15s use the non-HQ-II frequency when specified aircraft that were not HQ-II capable flew in the TAOR. One factor involved in the accident was that Black Hawk helicopters (UH-60s) were *not* on the list of non-HQ-II aircraft that must be contacted using a non-HQ-II mode.

Identification of aircraft was assisted by electronic Air-to-Air Interrogation/Identification Friend or Foe (AAI/IFF) systems. Each coalition aircraft was equipped with an IFF transponder. Friendly radars (located in the AWACS, a fighter aircraft, or a ground site) execute what is called a *parrot check* to determine if the target being reflected on their radar screens is friendly or hostile. The AAI component (the interrogator) sends a signal to an airborne aircraft to determine its identity, and the IFF component answers or *squawks back* with a secret code—a numerically identifying pulse that changes daily and must be uploaded into aircraft using secure equipment prior to takeoff. If the return signal is valid, it appears on the challenging aircraft's visual display (radarscope). A compatible code has to be loaded into the cryptographic system of both the challenging and the responding aircraft to produce a friendly response.

An F-15's AAI/IFF system can interrogate using four identification signals or modes. The different types of IFF signals provide a form of redundancy. Mode I is a general identification signal that permits selection of 32 codes. Two Mode I codes were designated for use in OPC at the time of the accident: one for inside the TAOR and the other for outside. Mode II is an aircraft-specific identification mode allowing the use of 4,096 possible codes. Mode III provides a nonsecure friendly identification of both military and civilian aircraft and was not used in the TAOR. Mode IV is secure and provides high-confidence identification of friendly targets. According to the ACO, the primary means of identifying friendly aircraft in the Iraqi no-fly-zone were to be modes I and IV in the IFF interrogation process.

Physical identification is also important in preventing friendly fire accidents. The ROE require that the pilots perform a visual identification of the potential threat. To assist in this identification, the Black Hawks were marked with six two-by-three foot American flags. An American flag was painted on each door, on both sponsons,[5] on the nose, and on the belly of each helicopter [90]. A flag had been added to the side of each sponson because the Black Hawks had been the target of small arms ground fire several months before.

Safety Constraints Related to the Accident:

1. Pilots must know and follow the rules of engagement established and communicated from the levels above,

2. Pilots must know who is in the No-Fly-Zone at all times and whether they should be there or not, i.e., they must be able to accurately identify the status of all other aircraft in the No-Fly-Zone at all times and must not misidentify a friendly aircraft as a threat.

3. Pilots of aircraft in the area must be able to hear radio communications.

4. Fixed-wing aircraft must fly above 10,000 feet and helicopters must remain below 400 feet.

Controls: Controls included the ACO, the ATO, flowsheets, radios, IFF, the ROE, training, the AWACS, procedures to keep fighters and helicopters from coming into contact (e.g., the F-15s fly at different altitudes), and special tactical radio frequencies when operating in the TAOR. Flags were displayed prominently on all aircraft in order to identify their origin.

With all these controls and this elaborate control structure to protect against friendly fire accidents, how could the shootdown occur on a clear day with all equipment operational? The next chapter contains an analysis of this accident using STAMP to answer these questions. But first, the rest of STAMP needs to be described.

[5] Sponsons are auxiliary fuel tanks

Chapter 7

Accident Factors

In STAMP, accidents result from inadequate enforcement of safety constraints on the behavior of the system components—the control loops between the various levels of the hierarchical control structure create or do not handle dysfunctional interactions leading to violations of the safety constraints. Dysfunctional interactions may result both from component failures and system design flaws.

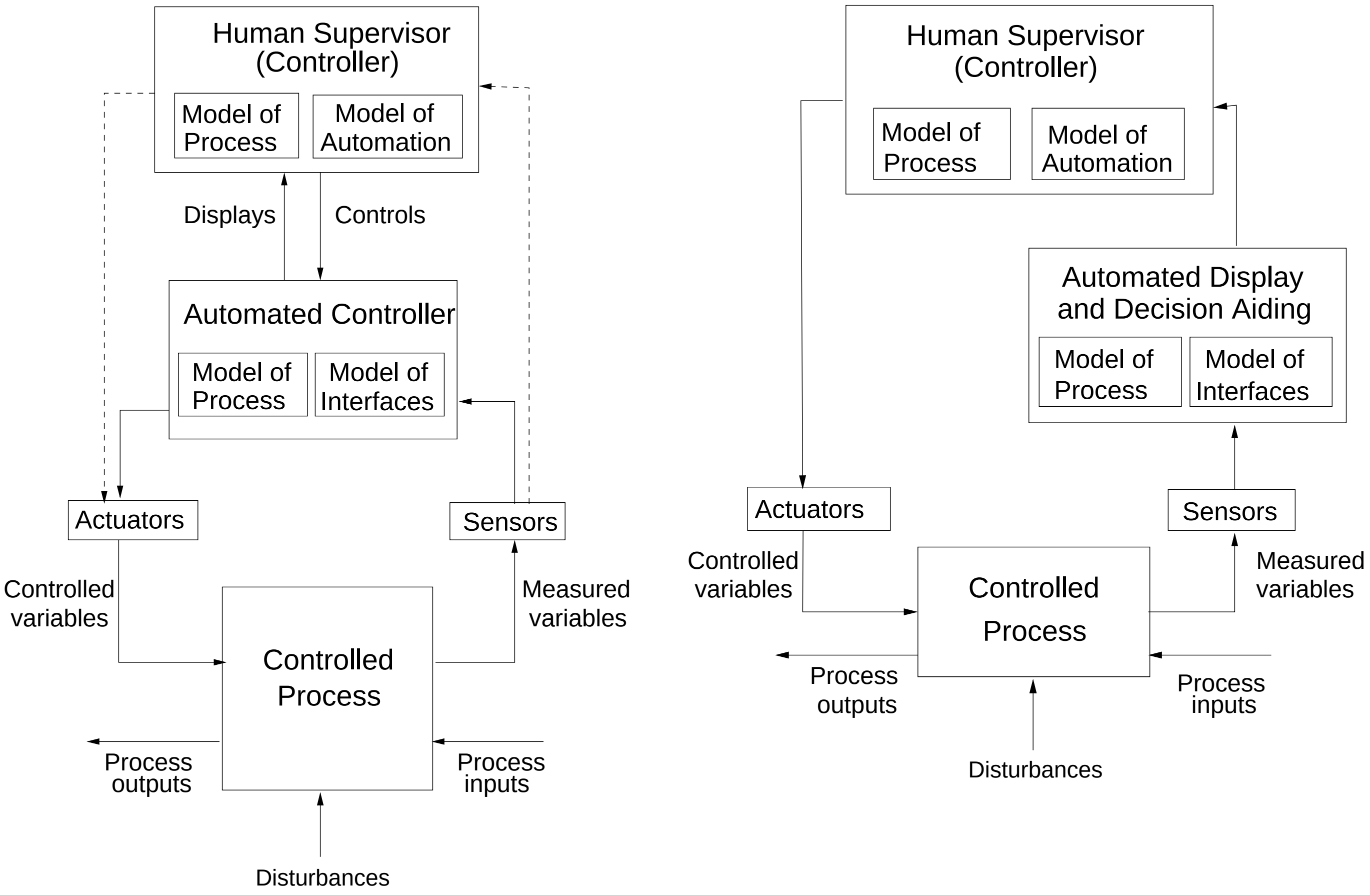
Starting from this basic definition of an accident, the process that leads to a loss can be understood in terms of flaws in the components of the system development and systems operations control loops in place during design, development, manufacturing, and operations. This chapter presents a classification of those flaws. The classification can be used during accident analysis or accident prevention activities to assist in identifying the factors involved in an accident and the relationships among them.

Process models (the third component of STAMP, along with constraints and hierarchical levels of control) play an important role in this classification and need to be discussed before the classification is presented.

7.1 The Role of Process Models in System Accidents

The importance of the human operator's mental model of the controlled process was discussed in Section 3.3. More generally, *any* controller—human or automated—needs a model of the process being controlled to effectively control it (the *model condition* discussed in Section 4.3 and in [20]). At one extreme, this process model may contain only one or two variables, such as the model required for a simple thermostat, which contains the current temperature and the setpoint. At the other extreme, effective control may require a complex model with a large number of state variables and transitions, such as the model needed to control air traffic.

Whether the model is embedded in the control logic of an automated controller or in the mental model maintained by a human controller, it must contain the same type of information: the required relationship among the system variables (the control laws), the current state (the current values of the system variables), and the ways the process can change state. This model is used to determine what control actions are needed, and it is updated through various forms of feedback. Note that a model of the process being controlled is required at all levels of the hierarchical control structure, not simply the lowest technical level.



(a.) Automation directly issuin g commands with supervision by a human controlle r.

(b.) Human control with automat ed assistance.

Figure 7.1: Accidents occur when the internal models of the controllers become inconsistent and do not match the system being controlled.

Figure 7.1a shows a typical process-control loop with an automated controller supervised by a human controller. The dashed lines indicate that the human supervisor may have direct access to system state information (not coming through the computer) and may have ways to manipulate the controlled process state other than through computer commands.

There may, of course, be multiple human and automated controllers in the control loop, and computers may be in other parts of the control loop. For example, computers may act as automated decision aids that provide information to the human controller but do not directly issue control commands to the process actuators: Figure 7.1b shows an example. If the software provides decision aiding, then it must contain a model of the process because it is indirectly controlling the process. Common arguments that in this design the software is not safety-critical are not justified—it is still a critical part of the functioning of the control loop and its behavior will affect the human controllers' decisions and actions.

This discussion has been simplified by speaking only of process models. Models will also need to include the relevant properties of the sensors, actuators, and some aspects of the environment, particularly those related to any potential environmental disturbances. An example is the need for an automated controller to have a model of its interface to the human controller(s) or supervisor(s). This interface, which contains the controls, displays, alarm annunciators, etc., is important because it is the means by which the automated and human controller models are synchronized, and lack of synchronization between the models and the inconsistencies that result often result in system accidents.

Note that process models are not only important during operations, but they are used during system development activities. Designers use both models of the system being designed and models of the development process itself. As an example of the latter, a loss of a Titan/Centaur spacecraft involved flaws in the developers' models of the testing process—each thought someone else was testing the software using the actual load tape when, in fact, nobody was.

7.2 A Classification of Accident Factors using STAMP

Now that the primary components of STAMP have been introduced—constraints, hierarchical control structures, and process models—a classification of accident factors can be derived for the model (Figure 7.2).

In STAMP, accidents are defined in terms of violations of safety constraints, which may result from system component failure(s), environmental disturbances, and dysfunctional interactions among (failing or nonfailing) components. In each control loop at each level of the socio-technical control structure for a particular system, unsafe behavior results from either a missing or inadequate constraint on the process at the lower level or inadequate enforcement of the constraint leading to its violation.

Because each component of the control loop may contribute to inadequate enforcement of safety constraints, classification starts by examining each of the general control-loop components and evaluating their potential contribution: (1) the controller may issue inadequate or inappropriate control actions, including inadequate handling of failures or disturbances in the physical process or (2) control actions may be inadequately executed by the actuator. These same general factors apply at each level of the socio-technical control structure, but the interpretations (applications) of the factor at each level may differ.

1. Inadequate Enforcement of Constraints (Control Actions)

- 1.1 Unidentified hazards
- 1.2 Inappropriate, ineffective, or missing control actions for identified hazards
  - 1.2.1 Design of control algorithm (process) does not enforce constraints
    - – Flaw(s) in creation process
    - – Process changes without appropriate change in control algorithm (asynchronous evolution)
    - – Incorrect modification or adaptation
  - 1.2.2 Process models inconsistent, incomplete, or incorrect (lack of linkup)
    - – Flaw(s) in creation process
    - – Flaws(s) in updating process
    - – Inadequate or missing feedback
      - Not provided in system design
      - Communication flaw
      - Time lag
      - Inadequate sensor operation (incorrect or no information provided)
    - – Time lags and measurement inaccuracies not accounted for
  - 1.2.3 Inadequate coordination among controllers and decision makers (boundary and overlap areas)

2. Inadequate Execution of Control Action

- 2.1 Communication flaw
- 2.2 Inadequate actuator operation
- 2.3 Time lag

Figure 7.2: A Classification of Control Flaws Leading to Hazards

For each of the factors, at any point in the control loop where a human or organization is involved, it will be necessary to evaluate the context in which decisions are made and the behavior-shaping mechanisms (influences) at play in order to understand how and why unsafe decisions have been made.

Note that accidents caused by basic component failures are included here. Component failures may result from inadequate constraints on the manufacturing process; inadequate engineering design such as missing or incorrectly implemented fault tolerance; lack of correspondence between individual component capacity (including humans) and task requirements; unhandled environmental disturbances (e.g., EMI); inadequate maintenance, including preventive maintenance; physical degradation over time (wearout), etc.

Component failures may be prevented by increasing the integrity or resistance of the component to internal or external influences or by building in safety margins or safety factors. They may also be avoided by operational controls, such as operating the component within its design envelope and by periodic inspections and preventive maintenance. Manufacturing controls can reduce deficiencies or flaws introduced during the manufacturing process. The effects of component failure on system behavior may be eliminated or reduced by using redundancy. STAMP goes beyond simply blaming component failure for accidents and requires that the reasons be identified for why those failures occurred and led to an accident.

7.2.1 Inadequate Enforcement of Safety Constraints

Inadequate control over (enforcement of) safety constraints, can occur either because hazards and their related constraints were not identified (1.1 in Figure 7.2) or because the control actions do not adequately enforce the constraints (1.2). The latter may, in turn, result from flawed control algorithms (1.2.1), inconsistent or incorrect process models used by the control algorithms (1.2.2), inadequate coordination among multiple controllers and decision makers (1.2.3), or inadequate or missing feedback (1.2.4).

Inadequate Control Algorithms: Control algorithms may not enforce safety constraints (1.2.1) because the algorithms are inadequately designed originally, the process may change and thus the algorithms become inadequate, or they may be inadequately modified by maintainers (if the algorithms are automated) or through various types of natural adaptation if they are implemented by humans. Leplat has noted that many accidents relate to *asynchronous evolution* [63] where one part of a system (in our case the hierarchical safety control structure) changes without the related necessary changes in other parts. Changes to subsystems may be carefully designed, but consideration of their effects on other parts of the system, including the safety control aspects, may be neglected or inadequate. Asynchronous evolution may also occur when one part of a properly designed system deteriorates. In both these cases, the erroneous expectations of users or system components about the behavior of the changed or degraded subsystem may lead to accidents. The Ariane 5 trajectory changed from that of the Ariane 4, but the inertial reference system software did not. One factor in the loss of contact with SOHO (SOlar Heliosperic Observatory) in 1998 was the failure to communicate to operators that a functional change had been made in a procedure to perform gyro spin down. The Black Hawk friendly fire accident had several examples of asynchronous evolution, for example the CTF mission changed and an individual key to communication between the Air Force and Army left, without his duties being assigned to someone else.

Communication is a critical factor here as well as monitoring for changes that may occur and feeding back this information to the higher-level control. For example, the safety analysis process that generates constraints always involves some basic assumptions about the operating environment of the process. When the environment changes such that those assumptions are no longer true, the controls in place may become inadequate. Embedded pacemakers, for example, were originally assumed to be used only in adults, who would lie quietly in the doctor's office while the pacemaker was being "programmed." Later they began to be used in children, and the assumptions under which the hazard analysis was conducted and the controls were designed no longer held and needed to be revisited.

Inconsistent Process Models: Section 7.1 stated that effective control is based on a model of the process state. Accidents, particularly system accidents, most often result from inconsistencies between the models of the process used by the controllers (both human and automated) and the actual process state (1.2.2). When the controller's model of the process (either the human mental model or the software model) diverges from the process state, erroneous control commands (based on the incorrect model) can lead to an accident—for example, (1) the software does not know that the plane is on the ground and raises the landing gear or (2) the controller (automated or human) does not identify an object as friendly and shoots a missile at it or (3) the pilot thinks the aircraft controls are in *speed* mode but the computer has changed the mode to *open descent* and the pilot issues inappropriate commands for that mode or (4) the computer does not think the aircraft has landed and overrides the pilots' attempts to operate the braking system. All of these examples have actually occurred.

The mental models of the system developers are also important. During software development, for example, the programmers' models of required behavior may not match engineers' models (commonly referred to as software requirements error), or the software may be executed on computer hardware or may control physical systems during operations that differs from what was assumed by the programmer and used during testing. The situation becomes more even complicated when there are multiple controllers (both human and automated) because each of their process models must also be kept consistent.

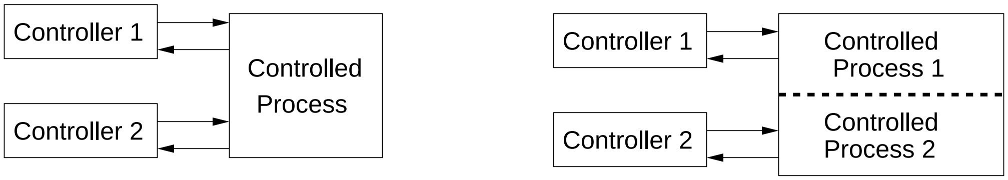
The most common form of inconsistency occurs when one or more of the process models is incomplete in terms of not defining appropriate behavior for all possible process states or all possible disturbances, including unhandled or incorrectly handled component failures. Of course, no models are complete in the absolute sense: The goal is to make them complete enough that no safety constraints are violated when they are used. Definitions of completeness in this sense are described in Chapter 15 of *Safeware* [66].

How do the models become inconsistent? First, they may be wrong from the beginning, such as incorrect system or software requirements). In this case, the design of the controller itself is flawed: there may be uncontrolled disturbances, unhandled process states, inadvertent commands of the system into a hazardous state, unhandled or incorrectly handled system component failures, etc.

Second, the models may start out being accurate but may become incorrect due to lack of feedback, inaccurate feedback, or inadequate processing of the feedback. A contributing factor cited in the Cali B-757 accident report was the omission of the waypoints behind the aircraft from cockpit displays, which contributed to the crew not realizing that the waypoint for which they were searching was behind them (missing feedback). The model of the Ariane 501 attitude used by the attitude control software became inconsistent with the launcher attitude when an error message sent by the inertial reference system was interpreted by the attitude control system as data (incorrect processing of feedback), leading to the issuance of an incorrect and unsafe control command.

Other reasons for the process models to diverge may be more subtle. Information about the process state has to be inferred from measurements. For example, in the TCAS II aircraft collision avoidance system, relative range positions of other aircraft are computed based on round-trip message propagation time. The theoretical control function (control law) uses the true values of the controlled variables or component states (e.g., true aircraft positions). However, at any time, the controller has only measured values, which may be subject to time lags or inaccuracies. The controller must use these measured values to infer the true conditions in the process and, if necessary, to derive corrective actions to maintain the required process state. In the TCAS example, sensors include on-board devices such as altimeters that provide measured altitude (not necessarily true altitude) and antennas for communicating with other aircraft. The primary TCAS actuator is the pilot, who may or may not respond to system advisories. The mapping between the measured or assumed values and the true values can be flawed.

In addition, the control loop must necessarily include time lags, such as the time between measuring values and receiving those values or between issuing a command and the actual process state change. Pilot response delays are important time lags that must be considered in designing the control function for TCAS or other aircraft systems as are time lags in the controlled process



(a) Example of an overlap

(b) Example of a boundary area

Figure 7.3: Problems often occur when there is shared control over the same process or at the boundary areas or separately controlled processes.

(the aircraft trajectory) caused by aircraft performance limitations. Delays may not be directly observable, but may need to be inferred. Depending on where in the feedback loop the delay occurs, different models are required to cope with the delays [15]: dead time and time constants require a model that makes it possible to predict when an action is needed before the need arises while feedback delays require a model that allows prediction of when a given action has taken effect and when resources will be available again. Such requirements may impose the need for some type of open loop or feedforward strategy to cope with delays.

To summarize, process models can be incorrect from the beginning (where correct is defined in terms of consistency with the current process state and with the models being used by other controllers) or they can become incorrect due to erroneous or missing feedback or measurement inaccuracies. They may also be incorrect only for short periods of time due to time lags in the process loop.

Inadequate Coordination Among Controllers and Decision Makers: When there are multiple controllers (human and/or automated), control actions may be inadequately coordinated (1.2.3), including unexpected side effects of decisions or actions or conflicting control actions. Communication flaws play an important role here.

Leplat suggests that accidents are most likely in *overlap areas* or in *boundary areas* or where two or more controllers (human and/or automated) control the same process or processes with common boundaries (Figure 7.3) [63]. In both boundary and overlap areas, the potential exists for ambiguity and for conflicts among independent decisions.

Responsibility for the control functions in boundary areas are often poorly defined. For example, Leplat cites an iron and steel plant where frequent accidents occurred at the boundary of the blast furnace department and the transport department. One conflict arose when a signal informing transport workers of the state of the blast furnace did not work and was not repaired because each department was waiting for the other to fix it. Faverge suggests that such dysfunctioning can be related to the number of management levels separating the workers in the departments from a common manager: The greater the distance, the more difficult the communication, and thus the greater the uncertainty and risk.

Coordination problems in the control of boundary areas are rife. As mentioned earlier, a Milstar satellite was lost due to inadequate attitude control of the Titan/Centaur launch vehicle, which used an incorrect process model based on erroneous inputs in a software load tape. After the accident, it was discovered that nobody had tested the software using the actual load tape—each group involved in testing and assurance had assumed some other group was doing so. In the system development process, system engineering and mission assurance activities were missing or ineffective, and a common control or management function was quite distant from the individual development and assurance groups. A factor in the loss of the Black Hawk helicopters to friendly fire over northern Iraq was that the helicopters normally flew only in the boundary areas of the No-Fly-Zone and procedures for handling aircraft in those areas were ill-defined. Another factor was that an Army base controlled the flights of the Black Hawks while an Air Force base controlled all the other components of the airspace. A common control point once again was high above where the accident occurred in the control structure. In addition, communication problems existed between the Army and Air Force bases at the intermediate control levels.

Overlap areas exist when a function is achieved by the cooperation of two controllers or when two controllers exert influence on the same object. Such overlap creates the potential for conflicting control actions (dysfunctional interactions among control actions). Leplat cites a study of the steel industry that found 67 percent of technical incidents with material damage occurred in areas of co-activity, although these represented only a small percentage of the total activity areas. In an A320 accident in Bangalore, India, the pilot had disconnected his flight director during approach and assumed that the co-pilot would do the same. The result would have been a mode configuration in which airspeed is automatically controlled by the autothrottle (the *speed* mode), which is the recommended procedure for the approach phase. However, the co-pilot had not turned off his flight director, which meant that *open descent* mode became active when a lower altitude was selected instead of *speed* mode, eventually contributing to the crash of the aircraft short of the runway [102]. In the Black Hawks' shootdown by friendly fire, the aircraft surveillance officer (ASO) thought she was responsible only for identifying and tracking aircraft south of the 36th parallel while the air traffic controller for the area north of the 36th parallel thought the ASO was also tracking and identifying aircraft in his area and acted accordingly.

Inadequate or Missing Feedback: The third flaw leading to system hazards involves inadequate feedback (1.2.4). A basic principle of system theory is that no control system will perform better than its measuring channel. Important questions therefore arise about whether the controllers or decision makers (either automated or human) have the necessary information about the actual state of the controlled process to satisfy their objectives. This information is contained in their process models and updating these models correctly is crucial to avoiding accidents (1.2.2). Feedback may be missing or inadequate because such feedback is not included in the system design, flaws exist in the monitoring or feedback communication channel, the feedback is not timely, or the measuring instrument operates inadequately.

7.2.2 Inadequate Execution of Control Actions

A second way for constraints to be violated in the controlled process is if there is a failure or inadequacy in the reference channel, i.e., in the transmission of control commands or in their execution, i.e., an actuator or controlled component fault or failure. Again, these types of flaws do not simply apply to operations or to the technical system but also to system design and development. For example, a common flaw in system development is that the safety information gathered or created by the system safety engineers (the hazards and the necessary design constraints to control them) is inadequately communicated to the system designers and testers or flaws exist in the application of this information in the system development process.

The next secion continues the friendly fire example, which incorporates almost all of these accident factors.

7.3 The Friendly Fire Example Continued

We can now return to the Black Hawk friendly fire accident as an example of the use of STAMP.

Chapter 6 described the hierarchical structure in place to prevent friendly fire accidents in the Iraqi No-Fly-Zone. As noted, frendly fire is well known as a hazard, and many controls were in place in OPC to prevent such an occurrence. As the Chairman of the Joint Chiefs of Staff said after the accident:

> In place were not just one, but a series of safeguards—some human, some procedural, some technical—that were supposed to ensure an accident of this nature could never happen. Yet, quite clearly, these safeguards failed.[1]

Understanding why this accident occurred and learning how to prevent such losses in the future requires determining why these safeguards were not successful in preventing the friendly fire. Various explanations for the accident have been posited. Making sense out of these conflicting explanations and understanding the accident process involved (including not only failures of individual system components but the dysfunctional interactions and miscommunications between components) requires understanding the role played in this process by each of the elements of the safety control structure in place at the time.

The next section contains a description of the proximate events involved in the loss. Then the STAMP model explaining these events is presented.

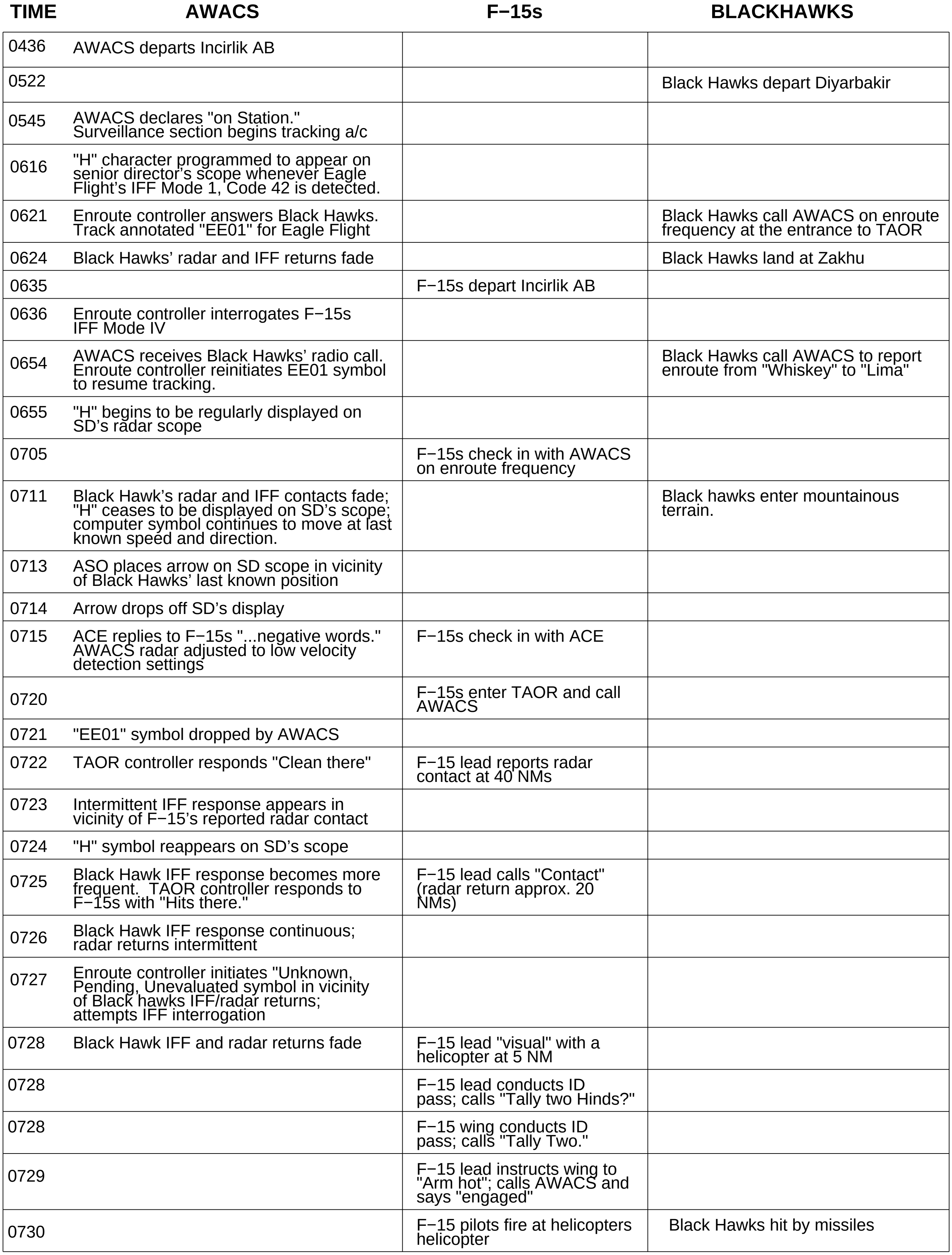
7.3.1 Some Proximate Events to the Loss

Table 7.4, taken from the official Accident Investigation Board Report, shows a time line of the actions of each of the main actors in the proximate events—the AWACS, the F-15s, and the Black Hawks. It may also be helpful to refer back to Figure 6.3, which contains a map of the area showing the relative locations of the important activities.

After receiving a briefing on the day's mission, the AWACS took off from Incirlik Air Base. When they arrived on station and started to track aircraft, the AWACS surveillance section noticed unidentified radar returns (from the Black Hawks). A "friendly general" track symbol was assigned to the aircraft and labeled as *H*, denoting a helicopter. The Black Hawks (Eagle Flight) later entered the TAOR[2] through Gate 1, checked in with the AWACS controllers who annotated the track with the identifier *EE01*, and flew to Zakhu. The Black Hawk pilots did not change their IFF (Identify Friend or Foe) Mode I code: The code for all friendly fixed-wing aircraft flying in Turkey on that day was 42 and the code for the TAOR was 52. They also remained on the enroute radio frequency instead of changing to the frequency to be used in the TAOR. When the helicopters landed at Zakhu, their radar and IFF (Identify Friend or Foe) returns on the AWACS radarscopes faded. Thirty minutes later, Eagle Flight reported their departure from Zakhu to the AWACS and

[1] John Shalikashvili, Chairman of the Joint Chiefs of Staff, from a cover letter to the twenty-one volume Report of the Aircraft Accident Investigation Board, 1994a, page 1.

[2] Tactical Area of Responsibility, the official name of the No-Fly-Zone.



TIME	AWACS	F−15s	BLACKHAWKS
0436	AWACS departs Incirlik AB		
0522			Black Hawks depart Diyarbakir
0545	AWACS declares "on Station." Surveillance section begins tracking a/c		
0616	"H" character programmed to appear on senior director's scope whenever Eagle Flight's IFF Mode 1, Code 42 is detected.		
0621	Enroute controller answers Black Hawks. Track annotated "EE01" for Eagle Flight		Black Hawks call AWACS on enroute frequency at the entrance to TAOR
0624	Black Hawks' radar and IFF returns fade		Black Hawks land at Zakhu
0635		F−15s depart Incirlik AB	
0636	Enroute controller interrogates F−15s IFF Mode IV		
0654	AWACS receives Black Hawks' radio call. Enroute controller reinitiates EE01 symbol to resume tracking.		Black Hawks call AWACS to report enroute from "Whiskey" to "Lima"
0655	"H" begins to be regularly displayed on SD's radar scope		
0705		F−15s check in with AWACS on enroute frequency	
0711	Black Hawk's radar and IFF contacts fade; "H" ceases to be displayed on SD's scope; computer symbol continues to move at last known speed and direction.		Black hawks enter mountainous terrain.
0713	ASO places arrow on SD scope in vicinity of Black Hawks' last known position		
0714	Arrow drops off SD's display		
0715	ACE replies to F−15s "...negative words." AWACS radar adjusted to low velocity detection settings	F−15s check in with ACE	
0720		F−15s enter TAOR and call AWACS	
0721	"EE01" symbol dropped by AWACS		
0722	TAOR controller responds "Clean there"	F−15 lead reports radar contact at 40 NMs	
0723	Intermittent IFF response appears in vicinity of F−15's reported radar contact		
0724	"H" symbol reappears on SD's scope		
0725	Black Hawk IFF response becomes more frequent. TAOR controller responds to F−15s with "Hits there."	F−15 lead calls "Contact" (radar return approx. 20 NMs)	
0726	Black Hawk IFF response continuous; radar returns intermittent		
0727	Enroute controller initiates "Unknown, Pending, Unevaluated symbol in vicinity of Black hawks IFF/radar returns; attempts IFF interrogation		
0728	Black Hawk IFF and radar returns fade	F−15 lead "visual" with a helicopter at 5 NM	
0728		F−15 lead conducts ID pass; calls "Tally two Hinds?"	
0728		F−15 wing conducts ID pass; calls "Tally Two."	
0729		F−15 lead instructs wing to "Arm hot"; calls AWACS and says "engaged"	
0730		F−15 pilots fire at helicopters helicopter	Black Hawks hit by missiles

Figure 7.4: The proximate chronological events leading to the accident.

said they were enroute from *Whiskey* (code name for Zakhu) to *Lima* (code name for Irbil, a town deep in the TAOR). The enroute controller reinitiated tracking of the helicopters.

Two F-15s were tasked that day to be the first aircraft in the TAOR and to *sanitize* it (check for hostile aircraft) before other coalition aircraft entered the area. The F-15s reached their final checkpoint before entering the TAOR approximately an hour after the helicopters had entered. They turned on all combat systems, switched their IFF Mode I code from 42 to 52, and switched to the TAOR radio frequency. They reported their entry into the TAOR to the AWACS.

At this point, the Black Hawks' radar and IFF contacts faded as the helicopters entered mountainous terrain. The AWACS computer continued to move the helicopter tracks on the radar display at the last known speed and direction, but the identifying *H* symbol (for helicopter) on the track was no longer displayed. The Air Surveillance Officer (ASO) placed an "attention arrow" (used to point out an area of interest) on the Senior Director's (SD) scope at the point of the Black Hawk's last known location. This large arrow is accompanied by a blinking alert light on the SD's console. The SD did not acknowledge the arrow and after sixty seconds, both the arrow and the light were automatically dropped. The ASO then adjusted the AWACS's radar to detect slow-moving objects.

Before entering the TAOR, the lead F-15 pilot checked in with the ACE and was told there were no relevant changes from previously briefed information ("negative words"). Five minutes later, the F-15's entered the TAOR and the lead pilot reported their arrival to the TAOR controller. One minute later, the enroute controller finally dropped the symbol for the helicopters from the scope, the last remaining visual reminder that there were helicopters inside the TAOR.

Two minutes after entering the TAOR, the lead F-15 picked up hits on its instruments indicating that it was getting radar returns from a low and slow-flying aircraft. The lead F-15 pilot alerted his wingman and then locked onto the contact and used the F-15's air-to-air interrogator to query the target's IFF code. If it was a coalition aircraft, it should be squawking Mode I, code 52. The scope showed it was not. He reported the radar hits to the controllers in the AWACS, and the TAOR controller told him they had no radar contacts in that location ("clean there"). The wing pilot replied to the wing pilot's alert, noting that his radar also showed the target.

The lead F-15 pilot then switched the interrogation to the second mode (Mode IV) that all coalition aircraft should be squawking. For the first second it showed the right symbol, but for the rest of the interrogation (4 to 5 seconds) it said the target was not squawking Mode IV. The lead F-15 pilot then made a second contact call to the AWACS over the main radio, repeating the location, altitude, and heading of his target. This time the AWACS enroute controller responded that he had radar returns on his scope at the spot ("hits there") but did not indicate that these returns might be from a friendly aircraft. At this point, the Black Hawk IFF response was continuous but the radar returns were intermittent. The enroute controller placed an "unknown, pending, unevaluated" track symbol in the area of the helicopter's radar and IFF returns and attempted to make an IFF identification.

The lead F-15 pilot, after making a second check of Modes I and IV and again receiving no response, executed a visual identification pass to confirm that the target was hostile (the next step required in the Rules of Engagement). He saw what he thought was an Iraqi helicopter. He pulled out his "goody book" with aircraft pictures in it, checked the silhouettes, and identified the helicopters as Hinds, a type of Russian aircraft flown by the Iraqis ("Tally two Hinds"). The F-15 wing pilot also reported seeing two helicopters ("Tally two"), but never confirmed that he had identified them as Hinds or as Iraqi aircraft.

The lead F-15 pilot called the AWACS and said they were engaging enemy aircraft ("Tiger

Two[3] has tallied two Hinds, engaged"), cleared his wingman to shoot ("Arm hot"), and armed his missiles. He then did one final Mode I check, received a negative response, and pressed the button that released the missiles. The wingman fired at the other helicopter and both were destroyed.

This description represents the chain of events, but it does not explain "why" the accident occurred except at the most superficial level and provides few clues as to how to redesign the system to prevent future occurrences. Just looking at these basic events surrounding the accident, it appears that mistakes verging on gross negligence were involved—undisciplined pilots shot down friendly aircraft in clear skies, and the AWACS crew and others who were supposed to provide assistance simply sat and watched without telling the F-15 pilots that the helicopters were there. An analysis using STAMP, as will be seen, provides a very different level of understanding. In the following analysis, the goal is to understand why the controls in place did not prevent the accident and to identify the changes necessary to prevent similar accidents in the future. A related type of analysis can be used during system design and development (see Chapter 9) to prevent such occurrences in the first place.

In the following analysis, the basic failures and dysfunctional interactions leading to the loss at the physical level are identified first. Then each level of the hierarchical safety control structure is considered in turn, starting from the bottom.

At each level, the context in which the behaviors took place is considered. The context for each level includes the hazards, the safety requirements and constraints, the controls in place to prevent the hazard, and aspects of the environment or situation relevant to understanding the control flaws, including the people involved, their assigned tasks and responsibilities, and any relevant environmental behavior-shaping factors. Following a description of the context, the dysfunctional interactions and failures at that level are described along with the accident factors (see Figure 7.2) that were involved.

7.3.2 Physical Process Failures and Dysfunctional Interactions

The first step in the analysis is to understand the physical failures and dysfunctional interactions within the physical process that were related to the accident. Figure 7.5 shows this information.

All the physical components worked exactly as intended, except perhaps for the IFF system. The fact that the Mode IV IFF gave an intermittent response has never been completely explained. Even after extensive equipment teardowns and reenactments with the same F-15s and different Black Hawks, no one has been able to explain why the F-15 IFF interrogator did not receive a Mode IV response [113]. The Accident Investigation Board report states that: "The reason for the unsuccessful Mode IV interrogation attempts cannot be established, but was probably attributable to one or more of the following factors: incorrect selection of interrogation modes, faulty air-to-air interrogators, incorrectly loaded IFF transponder codes, garbling of electronic responses, and intermittent loss of line-of-sight radar contact."[4]

There were several dysfunctional interactions and communication inadequacies among the correctly operating aircraft equipment. The most obvious dysfunctional interaction was the release of

[3] Tiger One was the code name for the F-15 lead pilot while Tiger Two denoted the wing pilot.

[4] The Commander, US Army in Europe objected to this sentence. He argued that nothing in the Board report supported the possibility that the codes had been loaded inproperly and that it was clear the Army crews were not at fault in this matter. The US Commander in Chief, Europe, agreed with his view. Although the language in the opinion was not changed, the Commander, US Army in Europe, said his concerns were addressed because the complaint had been included as an attachment to the Board report.

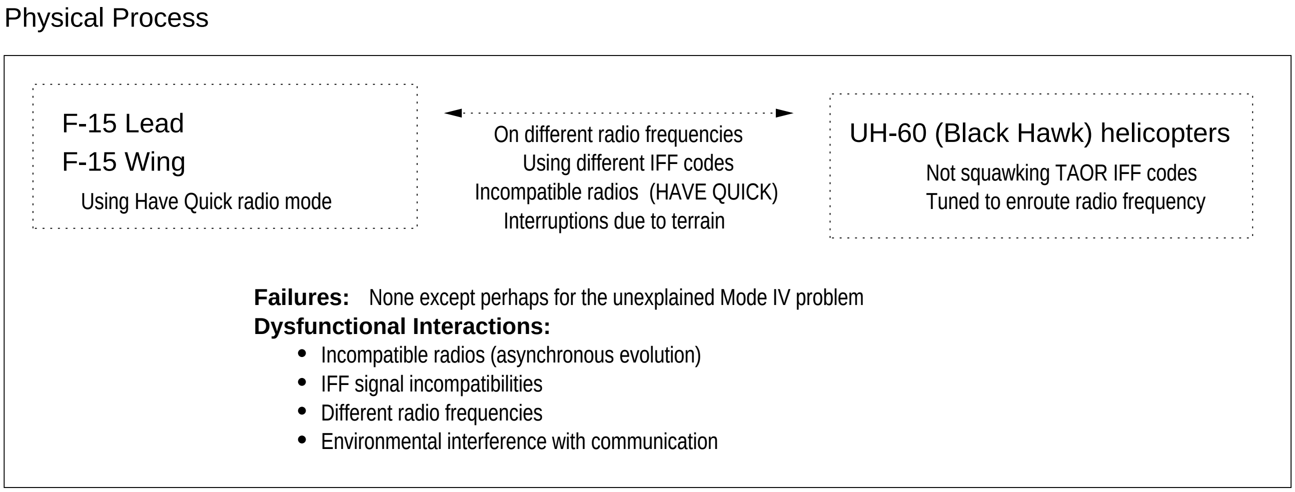

Physical Process
F-15 Lead
F-15 Wing
Using Have Quick radio mode
On different radio frequencies
Using different IFF codes
Incompatible radios (HAVE QUICK)
Interruptions due to terrain
UH-60 (Black Hawk) helicopters
Not squawking TAOR IFF codes
Tuned to enroute radio frequency
Failures: None except perhaps for the unexplained Mode IV problem
Dysfunctional Interactions:
Incompatible radios (asynchronous evolution)
IFF signal incompatibilities
Different radio frequencies
Environmental interference with communication

Figure 7.5: The Physical Level of the Accident Process.

two missiles in the direction of two friendly aircraft, but there were also four obstacles to the type of fighter-helicopter communications that might have prevented that release.

1. The Black Hawks and F-15s were on different radio frequencies and thus the pilots could not speak to each other or hear the transmissions between others involved in the incident, e.g., the radio transmissions between the two F-15 pilots and between the lead F-15 pilot and personnel onboard the AWACS. The Black Hawks, according to the Aircraft Control Order[5], should have been communicating on the TAOR frequency. Stopping here and looking only at this level, it appears that the Black Hawk pilots were at fault in not changing to the TAOR frequency, but an examination of the higher levels of control points to a different conclusion.

2. Even if they had been on the same frequency, the Air Force fighter aircraft were equipped with HAVE QUICK II (HQ-II) radios while the Army helicopters were not. The only way the F-15 and Black Hawk pilots could have communicated would have been if the F-15 pilots switched to non-HQ mode. The procedures the pilots were given to follow did not tell them to do so. In fact, with respect to the two helicopters that were shot down, one contained an outdated version called HQ-I, which was not compatible with HQ-II. The other *was* equipped with HQ-II, but because not all of the Army helicopters supported HQ-II, CFAC refused to provide Army helicopter operations with the necessary cryptographic support required to synchronize their radios with the other OPC components.

   If the objective of the accident analysis is to assign blame, then the different radio frequencies noted in Number 1 above could be considered irrelevant because the differing technology meant they could not have communicated even if they had been on the same frequency. If the objective, however, is to learn enough to prevent future accidents, then the different radio frequencies is relevant.

[5] The Aircraft Control Order or ACO contains the authoritative rules for all local air operations. All air crews are responsible for reviewing and complying with the information contained in the ACO.

3. The Black Hawks were not squawking the required IFF Mode I code for those flying within the TAOR. The GAO report states that Black Hawk pilots told them they routinely used the same Mode I code for outside the TAOR while operating within the TAOR and no one had advised them that it was incorrect to do so. But, again, the wrong Mode I code is only part of the story.

   The Accident Investigation Board report concluded that the use of the incorrect Mode I IFF code by the Black Hawks was responsible for the F-15 pilots' failure to receive a Mode I response when they interrogated the helicopters. However, an Air Force special task force concluded that based on the descriptions of the system settings that the pilots testified they had used on the interrogation attempts, the F-15s should have received and displayed any Mode I or II response *regardless of the code* [113]. The AWACS was receiving friendly Mode I and II returns from the helicopters at the same time that the F-15s received no response. The GAO report concluded that the helicopters' use of the wrong Mode I code should not have prevented the F-15s from receiving a response. Confusing the situation even further, the GAO report cites the Accident Board president as telling the GAO investigators that because of the difference between the lead pilot's statement on the day of the incident and his testimony to the investigation board, it was difficult to determine the number of times the lead pilot had interrogated the helicopters [113].

4. Communication was also impeded by physical line-of-sight restrictions. The Black Hawks were flying in narrow valleys among very high mountains that disrupted communication depending on line-of-sight transmissions.

One reason for these dysfunctional interactions lies in the *asynchronous evolution* of the Army and Air Force technology, leaving the different services with largely incompatible radios. Other reasons stem from inadequate control actions above this physical level, described later. Looking only at the event chain or at the failures and dysfunctional interactions in the technical process—a common stopping point in accident investigations—gives a very misleading picture of the reasons this accident occurred. Examining the higher levels of control is necessary to obtain the information necessary to prevent future occurrences.

After the shootdown, the following changes were made:

- Updated radios were placed on Black Hawk helicopters to enable communication with fighter aircraft. Until the time the conversion was complete, fighters were directed to remain on the TAOR clear frequencies for deconfliction with helicopters.

- Helicopter pilots were directed to monitor the common TAOR radio frequency and to squawk the TAOR IFF codes.

7.3.3 The Controllers of the Aircraft and Weapons

The pilots directly control the aircraft, including the activation of weapons. The context in which their decisions and actions took place is first described, followed by the disfunctional interactions at this level of the control structure. Then the inadequate control actions are outlined and the factors that led to them are described.

Context in Which Decisions and Actions Took Place

Safety Requirements and Constraints: The *safety constraints* that must be enforced at this level of the socio-technical control structure are described in Section 6.2. The F-15 pilots must know who is in the TAOR and whether they should be there or not, i.e., they must be able to accurately identify the status of all other aircraft in the TAOR at all times so that a friendly aircraft is not identified as a threat. They must also follow the rules of engagement (ROE), which specify the procedures to be executed before firing weapons at any targets. As noted in the previous chapter, the OPC ROE were devised by the OPC Commander, based on guidelines created by the Joint Chiefs of Staff, and were purposely conservative because of the many multinational participants in OPC and the potential for friendly fire accidents. The ROE were designed to slow down any military confrontation, but were unsuccessful in this case. An important part of understanding this accident process and preventing repetitions is understanding why this goal was not achieved.

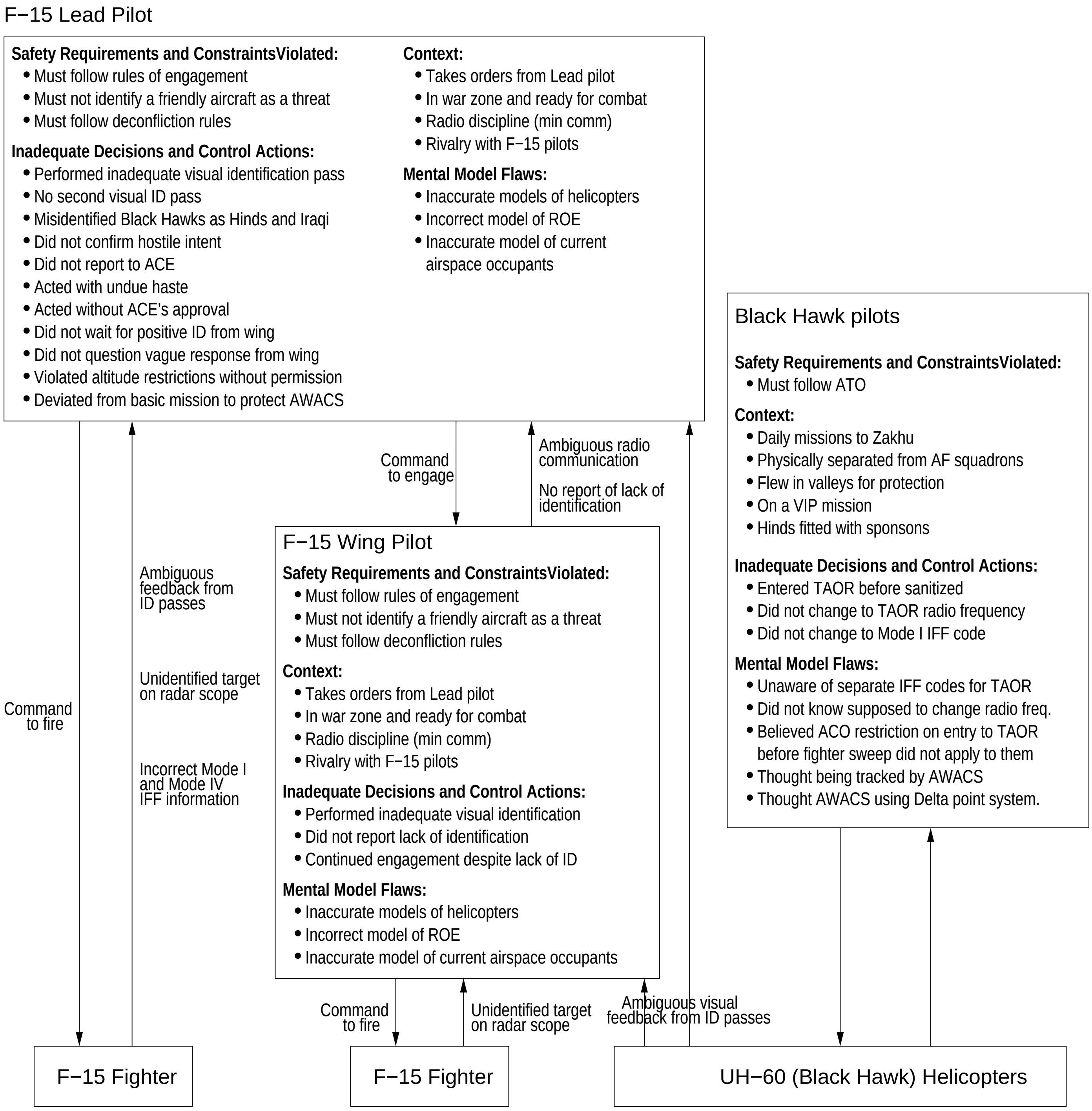
Controls: As noted in the previous chapter, the controls at this level included the rules and procedures for operating in the TAOR (specified in the ACO), information provided about daily operations in the TAOR (specified in the Air Tasking Order or ATO), flowsheets, communication and identification channels (radios and IFF), training, AWACS oversight, and procedures to keep fighters and helicopters from coming into contact (for example, the F-15s fly at different altitudes). National flags were required to be displayed prominently on all aircraft in order to facilitate identification of their origin.

Roles and Responsibilities of the F-15 Pilots: When conducting combat missions, aerial tactics dictate that F-15s always fly in pairs with one pilot as the lead and one as the wingman. They fly and fight as a team, but the lead is always in charge. The mission that day was to conduct a thorough radar search of the area to ensure that the TAOR was clear of hostile aircraft (to *sanitize* the airspace) before the other aircraft entered. They were also tasked to protect the AWACS from any threats. The wing pilot was responsible for looking 20,000 feet and higher with his radar while the lead pilot was responsible for the area 25,000 feet and below. The lead pilot had final responsibility for the 5,000 foot overlap area.

Environmental and Behavior-Shaping Factors for the F-15 Pilots: The lead pilot that day was a Captain with nine years experience in the AF. He had flown F-15s for over three years, including 11 combat missions over Bosnia and 19 over northern Iraq protecting the No-Fly-Zone. The mishap occurred on his sixth flight during his second tour flying in support of OPC.

The wing pilot was a lieutenant Colonel and 53rd Fighter Squadron Commander at the time of the shootdown and was a highly experienced pilot. He had flown combat missions out of Incirlik during Desert Storm and had served in the initial group that set up OPC afterward. He was credited with the only confirmed kill of an enemy Hind helicopter during the Gulf War. That downing involved a *beyond visual range* shot, which means he never actually saw the helicopter.

F-15 pilots were rotated through every 6 to 8 weeks. Serving in the No-Fly-Zone was an unusual chance for peacetime pilots to have a potential for engaging in combat. The pilots were very aware they were going to be flying in unfriendly skies. They drew personal sidearms with live rounds, removed wedding bands and other personal items that could be used by potential captors, were supplied with *blood chits* offering substantial rewards for returning downed pilots, and were briefed



F−15 Lead Pilot

Safety Requirements and ConstraintsViolated:
- Must follow rules of engagement
- Must not identify a friendly aircraft as a threat
- Must follow deconfliction rules

Inadequate Decisions and Control Actions:
- Performed inadequate visual identification pass
- No second visual ID pass
- Misidentified Black Hawks as Hinds and Iraqi
- Did not confirm hostile intent
- Did not report to ACE
- Acted with undue haste
- Acted without ACE's approval
- Did not wait for positive ID from wing
- Did not question vague response from wing
- Violated altitude restrictions without permission
- Deviated from basic mission to protect AWACS

Context:
- Takes orders from Lead pilot
- In war zone and ready for combat
- Radio discipline (min comm)
- Rivalry with F−15 pilots

Mental Model Flaws:
- Inaccurate models of helicopters
- Incorrect model of ROE
- Inaccurate model of current airspace occupants

F−15 Wing Pilot

Safety Requirements and ConstraintsViolated:
- Must follow rules of engagement
- Must not identify a friendly aircraft as a threat
- Must follow deconfliction rules

Context:
- Takes orders from Lead pilot
- In war zone and ready for combat
- Radio discipline (min comm)
- Rivalry with F−15 pilots

Inadequate Decisions and Control Actions:
- Performed inadequate visual identification
- Did not report lack of identification
- Continued engagement despite lack of ID

Mental Model Flaws:
- Inaccurate models of helicopters
- Incorrect model of ROE
- Inaccurate model of current airspace occupants

Black Hawk pilots

Safety Requirements and ConstraintsViolated:
- Must follow ATO

Context:
- Daily missions to Zakhu
- Physically separated from AF squadrons
- Flew in valleys for protection
- On a VIP mission
- Hinds fitted with sponsons

Inadequate Decisions and Control Actions:
- Entered TAOR before sanitized
- Did not change to TAOR radio frequency
- Did not change to Mode I IFF code

Mental Model Flaws:
- Unaware of separate IFF codes for TAOR
- Did not know supposed to change radio freq.
- Believed ACO restriction on entry to TAOR before fighter sweep did not apply to them
- Thought being tracked by AWACS
- Thought AWACS using Delta point system.

Command to engage
Ambiguous radio communication
No report of lack of identification
Ambiguous feedback from ID passes
Unidentified target on radar scope
Incorrect Mode I and Mode IV IFF information
Command to fire
Command to fire
Unidentified target on radar scope
Ambiguous visual feedback from ID passes
F−15 Fighter
F−15 Fighter
UH−60 (Black Hawk) Helicopters

Figure 7.6: The Analysis at the Pilot level.

about threats in the area. Every part of their preparation that morning drove home the fact that they could run into enemy aircraft. Thus the pilots were making decisions in the context of being in a war zone and were ready for combat.

Another factor that might have influenced behavior, according to the GAO report, was rivalry between the F-15 and F-16 pilots engaged in Operation Provide Comfort (OPC). While such rivalry was normally perceived as healthy and leading to positive professional competition, at the time of the shootdown the rivalry had become more pronounced and intense. The Combined Task Force Commander attributed this atmosphere to the F-16 community's having executed the only fighter shootdown in OPC and all the shootdowns in Bosnia [113]. F-16 pilots are better trained and equipped to intercept low-flying helicopters. The F-15 pilots knew that F-16s would follow them into the TAOR that day. Any hesitation might have resulted in the F-16s getting another kill.

A final factor was a strong cultural norm of "radio discipline" (called *minimum communication* or *min comm*), which led to abbreviated phraseology in communication and a reluctance to clarify potential miscommunications. Fighter pilots are kept extremely busy in the cockpit; their cognitive capabilities are often stretched to the limit. As a result, any unnecessary interruptions on the radio are a significant distraction from important competing demands [108]. Hence, there was a great deal of pressure within the fighter community to minimize talking on the radio, which discouraged efforts to check accuracy and understanding.

Roles and Responsibilities of the Black Hawk Pilots: The Army helicopter pilots flew daily missions into the TAOR to visit Zakhu. On this particular day, a change of command had taken place at the US Army Command Center at Zakhu. The outgoing commander was to escort his replacement into the No-Fly-Zone in order to introduce him to the two Kurdish leaders who controlled the area. The pilots were first scheduled to fly the routine leg into Zakhu, where they would pick up two Army colonels and carry other high-ranking VIPs representing the major players in OPC to the two Iraqi towns of Irbil and Salah ad Din. It was not uncommon for the Black Hawks to fly this far into the TAOR; they had done it frequently during the three preceding years of Operation Provide Comfort.

Environmental and Behavior-Shaping Factors for the Black Hawk Pilots: Inside Iraq, helicopters flew in terrain flight mode, i.e., they hugged the ground, both to avoid mid-air collisions and to mask their presence from threatening ground-to-air Iraqi radars. There are three types of terrain flight: Pilots select the appropriate mode based on a wide range of tactical and mission-related variables. *Low-level* terrain flight is flown when enemy contact is not likely. *Contour* flying is closer to the ground than low level, and *nap-of-the-earth* flying is the lowest and slowest form of terrain flight, flown only when enemy contact is expected. Eagle Flight helicopters flew contour mode most of the time in northern Iraq. They liked to fly in the valleys and the low-level areas. The route they were taking the day of the shootdown was through a green valley between two steep, rugged mountains. The mountainous terrain provided them with protection from Iraqi air defenses during the one-hour flight to Irbil, but it also led to disruptions in communication.

Because of the distance and thus time required for the mission, the Black Hawks were fitted with *sponsons* or pontoon-shaped fuel tanks. The sponsons are mounted below the side doors, and each hold 230 gallons of extra fuel. The Black Hawks were painted with green camouflage while the Iraqi Hinds' camouflage scheme was light brown and desert tan. To assist with identification, the Black Hawks were marked with three two-by-three foot American flags—one on each door and one on the nose—and a fourth larger flag on the belly of the helicopter. In addition, two American flags had been painted on the side of each sponson after the Black Hawks had been the target of small-arms ground fire several months before.

Dysfunctional Interactions at this Level

Communication between the F-15 and Black Hawk pilots was obviously dysfunctional and related to the dysfunctional interactions in the physical process (incompatible radio frequencies, IFF codes, and anti-jamming technology) resulting in the ends of the communication channels not matching and information not being transmitted along the channel. Communication between the F-15 pilots was also hindered by the *minimum communication* policy that led to abbreviated messages and a reluctance to clarify potential miscommunications as described above as well as the physical terrain.

Flawed or Inadequate Decisions and Control Actions

Both the Army helicopter pilots and the F-15 pilots executed inappropriate or inadequate control actions during their flights, beyond the obviously incorrect F-15 pilot commands to fire on two friendly aircraft.

Black Hawk Pilots:

- *The Army helicopters entered the TAOR before it had been sanitized by the Air Force.* The Air Control Order or ACO specified that a fighter sweep of the area must precede any entry of allied aircraft. However, because of the frequent trips of Eagle Flight helicopters to Zakhu, an official exception had been made to this policy for the Army helicopters. The Air Force fighter pilots had not been informed about this exception. Understanding this miscommunication requires looking at the higher levels of the control structure, particularly the communication structure at those levels.

- *The Army pilots did not change to the appropriate radio frequency to be used in the TAOR.* As noted above, however, even if they had been on the same frequency, they would have been unable to communicate with the F-15s because of the different anti-jamming technology of the radios.

- *The Army pilots did not change to the appropriate IFF Mode I signal for the TAOR.* Again, as noted above, the F-15s should still have been able to receive the Mode I response.

F-15 Lead Pilot

The accounts of and explanation for the unsafe control actions of the F-15 pilots differ greatly among those who have written about the accident. Analysis is complicated by the fact that any statements the pilots made after the accident were likely to have been influenced by the fact that they were being investigated on charges of negligent homicide—their stories changed significantly over time. Also, in the excitement of the moment, the lead pilot did not make the required radio call to his wingman requesting that he turn on the HUD[6] tape, and he also forgot to turn on his own tape. Therefore, evidence about certain aspects of what occurred and what was observed is limited to pilot testimony during the post-accident investigations and trials.

[6] Heads Up Display

Complications also arise in determining whether the pilots followed the Rules of Engagement (ROE) specified for the No-Fly-Zone because the ROE are not public and the relevant section of the Accident Investigation Board Report is censored. Other sources of information about the accident, however, reference clear instances of pilot violations of the ROE.

The following inadequate decisions and control actions can be identified for the lead F-15 pilot:

- *He did not perform a proper visual ID as required by the ROE and did not take a second pass to confirm the identification.* F-15 pilots are not accustomed to flying close to the ground or to terrain. The lead pilot testified that because of concerns about being fired on from the ground and the danger associated with flying in a narrow valley surrounded by high mountains, he had remained high as long as possible and then dropped briefly for a visual identification that lasted between 3 and 4 seconds. He passed the helicopter on his left while flying over 500 miles an hour and at a distance of about 1000 feet off to the side and about 300 feet above the helicopter. He testified that

  > "I was trying to keep my wing tips from hitting mountains and I accomplished two tasks simultaneously, making a call on the main radio and pulling out a guide that had the silhouettes of helicopters. I got only three quick interrupted glances of less than 1.25 seconds each" [90].

  The dark green Black Hawk camouflage blended into the green background of the valley, adding to the difficulty of the identification.

  The Accident Investigation Board used pilots flying F-15s and Black Hawks to recreate the circumstances under which the visual identification was made. The test pilots were unable to identify the Black Hawks, and they could not see any of the six American flags on each helicopter. The F-15 pilots could not have satisfied the ROE identification requirements using the type of visual identification passes they testified that they made.

- *He misidentified the helicopters as Iraqi Hinds*: There were two basic incorrect decisions involved in this misidentification. The first is identifying the UH-60 (Black Hawk) helicopters as Russian Hinds, and the second is assuming the Hinds were Iraqi. Both Syria and Turkey flew Hinds, and the helicopters could have belonged to one of the U.S. coalition partners. The Commander of the Operations Support Squadron, whose job was to run the weekly detachment squadron meetings, testified that as long as he had been in OPC, he had reiterated to the squadrons each week that they should be careful about misidentifying aircraft over the No-Fly-Zone because there were so many nations and so many aircraft in the area and that any time F-15s or anyone else picked up a helicopter on radar, it was probably a U.S., Turkish, or United Nations helicopter:

  > "Any time you intercept a helicopter as an unknown, there is always a question of procedures, equipment failure, and high terrain masking the line-of-sight radar. There are numerous reasons why you would not be able to electronically identify a helicopter. Use discipline. It is better to miss a shot than be wrong" [90].

- *He did not confirm, as required by the ROE, that the helicopters had hostile intent before firing*: The ROE required that the pilot not only determine the type of aircraft and nationality, but to take into consideration the possibility the aircraft was lost, in distress, on a medical mission, or was possibly being flown by pilots who were defecting.

- *He violated the Rules of Engagement by not reporting to the Air Command Element (ACE)*: According to the ROE, the pilot should have reported to the ACE (who is in his chain of command and physically located in the AWACS) that he had encountered an unidentified aircraft. He did not wait for the ACE to approve the release of the missiles.

- *He acted with undue and unnecessary haste that did not allow time for those above him in the control structure (who were responsible for controlling the engagement) to act*: The entire incident, from the first time the pilots received an indication about helicopters in the TAOR to shooting them down lasted only seven minutes. Pilots are allowed by the ROE to take action on their own in an emergency, so the question then becomes whether this situation was such an emergency.

  CFAC officials testified that there had been no need for haste. The slow-flying helicopters had traveled less than fourteen miles since the F-15s first picked them up on radar, they were not flying in a threatening manner, and they were flying southeast away from the Security Zone. The GAO report cites the Mission Director as stating that given the speed of the helicopters, the fighters had time to return to Turkish airspace, refuel, and still return and engage the helicopters before they could have crossed south of the 36th parallel.

  The helicopters also posed no threat to the F-15s or to their mission (which was to protect the AWACS and determine whether the area was clear). One expert later commented that even if they *had* been Iraqi Hinds,

  > “A Hind is only a threat to an F-15 if the F-15 is parked almost stationary directly in front of it and says ‘Kill me.’ Other than that, it’s probably not very vulnerable” [108].

  Piper quotes Air Force Lt. Col. Tony Kern, a professor at the U.S. Air Force Academy, who wrote about this accident:

  > “Mistakes happen, but there was no rush to shoot these helicopters. The F-15s could have done multiple passes, or even followed the helicopters to their destination to determine their intentions” [90].

  Any explanation behind the pilot’s hasty action can only be the product of speculation. Snook attributes the fast reaction to the overlearned defensive responses taught to fighter pilots. Both Snook and the GAO report mention the rivalry with the F-16 pilots and a desire of the lead F-15 pilot to shoot down an enemy aircraft. F-16s would have entered the TAOR 10 to 15 minutes after the F-15s, potentially allowing the F-16 pilots to get credit for the downing of an enemy aircraft: F-16s are better trained and equipped to intercept low-flying helicopters. If the F-15 pilots had involved the chain of command, the pace would have slowed down, ruining the pilots’ chance for a shootdown. In addition, Snook argues that this was a rare opportunity for peacetime pilots to engage in combat.

  The goals and motivation behind any human action are unknowable. Even in this case where the F-15 pilots survived the accident, there are many reasons to discount their own explanations, not the least of which is potential jail sentences. The explanations provided by the pilots right after the engagement differ significantly from their explanations a week later during the official investigations to determine whether they should be court martialed. But in any case, there was no chance that such slow flying helicopters could have escaped two supersonic jet fighters in the open terrain of northern Iraq nor were they ever a serious threat to the F-15s so this situation was not an emergency.

- *He did not wait for a positive ID from the wing pilot before firing on the helicopters and did not question the vague response when he got it*: When the lead pilot called out that he had visually identified two Iraqi helicopters, he asked the wing pilot to confirm the identification. The wingman called out "Tally Two" on his radio, which the lead pilot took as confirmation, but which the wing pilot later testified only meant he saw two helicopters but not necessarily Iraqi Hinds. The lead pilot did not wait for a positive identification from the wingman before starting the engagement.

- *He violated altitude restrictions without permission*: According to Piper, the commander of the OPC testified at one of the hearings that

  > "I regularly, routinely imposed altitude limitations in northern Iraq. On the fourteenth of April, the restrictions were a minimum of ten thousand feet for fixed-wing aircraft. This information was in each squadron's Aircrew Read File. Any exceptions had to have my approval" [90].

  None of the other accident reports, including the official one, mentions this erroneous action on the part of the pilots. Because this control flaw was never investigated, it is not possible to determine whether the action resulted from a "reference channel" problem (i.e., the pilots did not know about the altitude restriction) or an "actuator" error (i.e., the pilots knew about it but chose to ignore it.)

- *He deviated from the basic mission to protect the AWACS, leaving the AWACS open to attack*: The helicopter could have been a diversionary ploy. The mission of the first flight into the TAOR was to make sure it was safe for the AWACS and other aircraft to enter the restricted operating zone. Piper emphasizes that that was the only purpose of their mission [90]. Piper (who again is the only one who mentions it) cites testimony of the Commander of OPC during one of the hearings when asked whether the F-15s exposed the AWACS to other air threats when they attacked and shot down the helicopters. The Commander replied:

  > "Yes, when the F-15s went down to investigate the helicopters, made numerous passes, engaged the helicopters and then made more passes to visually reconnaissance the area, AWACS was potentially exposed for that period of time" [90].

Wing Pilot

The wing pilot, like the lead pilot, violated altitude restrictions, and deviated from the basic mission. In addition:

- *He did not make a positive identification of the helicopters*: His visual identification was not even as close to the helicopters as the lead F-15 pilot, which was inadequate to recognize the helicopters, and the wing pilot's ID lasted only between 2 and 3 seconds. According to a

Washington Post article, he told investigators that he never clearly saw the helicopters before reporting "Tally Two." In a transcript of one of his interviews with investigators, he said:

> "I did not identify them as friendly; I did not identify them as hostile. I expected to see Hinds based on the call my flight leader had made. I didn't see anything that disputed that."

Although the wing had originally testified he could not identify the helicopters as Hinds, he reversed his statement between April and six months later when he testified at the hearing on whether to court martial him that "I could identify them as Hinds" [90]. There is no way to determine which of these contradictory statements is true.

Explanations for continuing the engagement without an identification could range from an inadequate mental model of the ROE, following the orders of the lead pilot and assuming that his identification had been proper, wanting the helicopters to be hostile, and any combination of these.

- *He did not tell the lead pilot that he had not identified the helicopters*: In the hearings to place blame for the shootdown, the lead pilot testified that he had radioed the wing pilot and said "Tiger One has tallied two Hinds, confirm." Both pilots agree to this point, but then the testimony becomes contradictory.

  The hearing in the fall of 1984 on whether the wing pilot should be charged with 26 counts of negligent homicide rested on the very narrow question of whether the lead pilot had called the AWACS announcing the engagement before or after the wing pilot responded to the lead pilot's directive to confirm whether the helicopters were Iraqi Hinds. The lead pilot testified that he had identified the helicopters as Hinds and then asked the wing to confirm the identification. When the wing responded with "Tally Two," the lead believed this response signaled confirmation of the identification. The lead then radioed the AWACS and reported "Tiger Two has tallied two Hinds, engaged." The wing pilot, on the other hand, testified that the lead had called the AWACS with the "engaged" message before he (the wing pilot) had made his "Tally Two" radio call to the lead. He said his "Tally Two" call was in response to the "engaged" call, not the "confirm" call and simply meant that he had both target aircraft in sight. He argued that once the engaged call had been made, he correctly concluded that an identification was no longer needed.

  The Fall 1994 hearing conclusion about which of these scenarios actually occurred is different than the conclusions in the official Air Force accident report and that of the hearing officer in another hearing. Again, it is not possible nor necessary to determine blame here or to determine exactly which scenario is correct to conclude that the communications were ambiguous. The minimum communication policy was a factor here as was probably the excitement of a potential combat engagement. Snook suggests that the expectations of what the pilots expected to hear resulted in a filtering of the inputs. Such filtering is a well-known problem in airline pilots' communications with controllers. The use of well-established phraseology is meant to reduce it. But the calls by the wing pilot were nonstandard. In fact, Piper notes that in pilot training bases and programs that train pilots to fly fighter aircraft since the shootdown, these radio calls are used as examples of "the poorest radio communications possibly ever given by pilots during a combat intercept" [90].

- *He continued the engagement despite the lack of an adequate identification*: Explanations for continuing the engagement without an identification could range from an inadequate mental model of the ROE, following the orders of the lead pilot and assuming that the lead pilot's identification had been proper, wanting the helicopters to be hostile, and any combination of these. With only his contradictory testimony, it is not possible to determine the reason.

Some Reasons for the Flawed Control Actions and Dysfunctional Interactions

The accident factors shown in Figure 7.2 can be used to provide an explanation for the flawed control actions. These factors here are divided into incorrect control algorithms, inaccurate mental models, poor coordination among multiple controllers, and inadequate feedback from the controlled process.

Incorrect Control Algorithms: The Black Hawk pilots correctly followed the procedures they had been given (see the discussion of the CFAC–MCC level later). These procedures were unsafe and were changed after the accident.

The F-15 pilots apparently did not execute their control algorithms (the procedures required by the Rules of Engagement) correctly, although the secrecy involved in the ROE make this conclusion difficult to prove. After the accident, the ROE were changed, but the exact changes made are not public.

Inaccurate Mental Models of the F-15 Pilots: There were many inconsistencies between the mental models of the Air Force pilots and the actual process state. First, they had an ineffective model of what a Black Hawk helicopter looked like. There are several explanations for this, including poor visual recognition training and the fact that Black Hawks with sponsons attached resemble Hinds. None of the pictures of Black Hawks on which the F-15 pilots had been trained had these wing-mounted fuel tanks. Additional factors include the speeds at which the F-15 pilots do their visual identification (VID) passes and the angle at which the pilots passed over their targets.

Both F-15 pilots received only limited visual recognition training in the previous four months, partly due to the disruption of normal training caused by their wing's physical relocation from one base to another in Germany. But the training was probably inadequate even if it had been completed. Because the primary mission of F-15s is air-to-air combat against other fast moving aircraft, most of the operational training is focused on their most dangerous and likely threats—other high-altitude fighters. In the last training before the accident, only five percent of the slides depicted helicopters. None of the F-15 intelligence briefings or training ever covered the camouflage scheme of Iraqi helicopters, which was light brown and desert tan (in contrast to the forest green camouflage of the Black Hawks).

Pilots are taught to recognize many different kinds of aircraft at high speeds using "beer shots," which are blurry pictures that resemble how the pilot might see those aircraft while in flight. The Air Force pilots, however, received very little training in the recognition of Army helicopters, which they rarely encountered because of the different altitudes at which they flew. All the helicopter photos they did see during training, which were provided by the Army, were taken from the ground—a perspective from which it was common for Army personnel to view them but not useful for a fighter pilot in flight above them. None of the photographs were taken from the above aft quadrant—the position from which most fighters would view a helicopter. Air Force visual recognition training and procedures were changed after this accident.

The F-15 pilots also had an inaccurate model of the current airspace occupants, based on the information they had received about who would be in the airspace that day and when. They assumed and had been told in multiple ways that they would be the first coalition aircraft in the TAOR:

- The ACO specified that no coalition aircraft (fixed or rotary wing) was allowed to enter the TAOR before it was sanitized by a fighter sweep.
- The daily ATO and ARF included a list of all flights scheduled to be in the TAOR that day. The ATO listed the Army Black Hawk flights only in terms of their call signs, aircraft numbers, type of mission (transport), and general route (from Diyarbakir to the TAOR and back to Diyarbakir). All departure times were listed "as required" and no helicopters were mentioned on the daily flow sheet. Pilots fly with the flow sheet on kneeboards as a primary reference during the mission. The F-15s were listed as the very first mission into the TAOR; all other aircraft were scheduled to follow them.
- During preflight briefings that morning, the ATO and flowsheet were reviewed in detail. No mention was made of any Army helicopter flights not appearing on the flow sheet.
- The Battle Sheet Directive (a handwritten sheet containing last minute changes to information published in the ATO and the ARF) handed to them before going to their aircraft contained no information about Black Hawk flights.
- In a radio call to the ground-based Mission Director just after engine start, the lead F-15 pilot was told that no new information had been received since the ATO was published.
- Right before entering the TAOR, the lead pilot checked in again, this time with the ACE in the AWACS. Again, he was not told about any Army helicopters in the area.
- At 1020, the lead pilot reported that they were on station. Usually at this time, the AWACS will give them a "picture" of any aircraft in the area. No information was provided to the F-15 pilots at this time, although the Black Hawks had already checked in with the AWACS on three separate occasions.
- The AWACS continued not to inform the pilots about Army helicopters during the encounter. The lead F-15 pilot twice reported unsuccessful attempts to identify radar contacts they were receiving, but in response they were not informed about the presence of Blackhawks in the area. After the first report, the TAOR controller responded with "Clean there," meaning he did not have a radar hit in that location. Three minutes later, after the second call, the TAOR controller replied "Hits there." If the radar signal had been identified as a friendly aircraft, the controller would have responded "Paint there."
- The IFF transponders on the F-15s did not identify the signals as from a friendly aircraft (as discussed earlier).

Various complex interpretations have been proposed for why the F-15 pilots' mental models of the airspace occupants were incorrect and not open to reexamination once they received conflicting input. The simplest explanation is that they believed what they were told. It is well known in cognitive psychology that mental models are slow to change, particularly in the face of ambiguous evidence like that provided in this case. When operators receive input about the state of the system being controlled, they will first try to fit that information into their current mental model and will find reasons to exclude information that does not fit. Because operators are continually testing their mental models against reality, the longer a model has been held and the more different sources of information that led to that incorrect model, the more resistant the models will be to change due to conflicting information, particularly ambiguous information. The pilots had been told repeatedly and by almost everyone involved that there were no friendly helicopters in the TAOR at that time.

The F-15 pilots also may have had a misunderstanding about (incorrect model of) the ROE and the procedures required when they detected an unidentified aircraft. The accident report says that the ROE were reduced in briefings and in individual crew members' understandings to a simplified form. This simplification led to some pilots not being aware of specific considerations required prior to engagement, including identification difficulties, the need to give defectors safe conduct, and the possibility of an aircraft being in distress and the crew being unaware of their position. On the other hand, there had been an incident the week before and the F-15 pilots had been issued an oral directive reemphasizing the requirement for fighter pilots to report to the ACE. That directive was the result of an incident on April 7 in which F-15 pilots had initially ignored directions from the ACE to "knock off" or stop an intercept with an Iraqi aircraft. The ACE overheard the pilots preparing to engage the aircraft and contacted them, telling them to stop the engagement because he had determined that the hostile aircraft was outside the No-Fly-Zone and because he was leery of a "bait and trap" situation.[7] The GAO report stated that CFAC officials told the GAO that the F-15 community was "very upset" about the intervention of the ACE during the knock-off incident and felt he had interfered with the carrying out of the F-15 pilots' duties [113]. As discussed in Chapter 2, there is no way to determine the motivation behind an individual's actions. Accident analysts can only present the alternative explanations.

Additional reasons for the lead pilot's incorrect mental model stem from ambiguous or missing feedback from the F-15 wing pilot, dysfunctional communication with the Black Hawks, and inadequate information provided over the reference channels from the AWACS and CFAC operations.

Inaccurate Mental Models of the Black Hawk Pilots: The Black Hawk control actions can also be linked to inaccurate mental models, i.e., they were unaware there were separate IFF codes for flying inside and outside the TAOR and that they were supposed to change radio frequencies inside the TAOR. As will be seen later, they were actually told *not* to change frequencies. They had also been told that the ACO restriction on the entry of allied aircraft into the TAOR before the fighter sweep did not apply to them—an official exception had been made for helicopters. They understood that helicopters were allowed inside the TAOR without AWACS coverage as long as they stayed inside the security zone. In practice, the Black Hawk pilots frequently entered the TAOR prior to AWACS and fighter support without incident or comment, and therefore it became accepted practice. In addition, because their radios were unable to pick up the HAVE QUICK communications between the F-15 pilots and between the F-15s and the AWACS, the Black Hawk pilots' mental models of the situation were incomplete. According to Snook, Black Hawk pilots testified during the investigation that

> "We were not integrated into the entire system. We were not aware of what was going on with the F-15s and the sweep and the refuelers and the recon missions and AWACS. We had no idea who was where and when they were there" [108].

[7] According to the GAO report, in such a strategy, a fighter aircraft is lured into an area by one or more enemy targets and then attacked by other fighter aircraft or surface-to-surface missiles.

Coordination among Multiple Controllers: At this level, each component (aircraft) had a single controller and thus coordination problems did not occur. They were rife, however, at the higher control levels.

Feedback from the Controlled Process: The F-15 pilots received ambiguous information from their visual identification pass. At the speeds and altitudes they were traveling, it is unlikely that they would have detected the unique Black Hawk markings that identified them as friendly. The mountainous terrain in which they were flying limited their ability to perform an adequate identification pass and the green helicopter camouflage added to the difficulty. The feedback from the wingman to the lead F-15 pilot was also ambiguous and was most likely misinterpreted by the lead pilot. Both pilots apparently received incorrect IFF feedback.

Changes After the Accident

After the accident, Black Hawk pilots were:

- Required to strictly adhere to their ATO published routing and timing.
- Not allowed to operate in the TAOR unless under positive control of AWACS. Without AWACS coverage, only administrative helicopter flights between Diyarbakir and Zakhu were allowed, provided they were listed on the ATO.
- Required to monitor the common TAOR radio frequency.
- Required to confirm radio contact with AWACS at least every 20 minutes unless they were on the ground.
- Required to inform AWACS upon landing. They must make mandatory radio calls at each enroute point.
- If radio contact could not be established, required to climb to line-of-sight with AWACS until contact is reestablished.
- Prior to landing in the TAOR (including Zakhu), required to inform the AWACS of anticipated delays on the ground that would preclude taking off at the scheduled time.
- Immediately after takeoff, required to contact the AWACS and reconfirm IFF Modes I, II and IV are operating. If they have either a negative radio check with AWACS or an inoperative Mode IV, they cannot proceed into the TAOR.

All fighter pilots were:

- Required to check in with the AWACS when entering the low altitude environment and remain on the TAOR clear frequencies for deconfliction with helicopters.
- Required to make contact with AWACS using UHF, HAVE QUICK, or UHF clear radio frequencies and confirm IFF Modes I, II, and IV before entering the TAOR. If there was either a negative radio contact with AWACS or an inoperative Mode IV, they could not enter the TAOR.

Finally, white recognition strips were painted on the Black Hawk rotor blades to enhance their identification from the air.

7.3.4 ACE and Mission Director

Context in which Decisions and Actions Took Place

Safety Requirements and Constraints: The ACE and Mission Director must follow the procedures specified and implied by the ROE, the ACE must ensure that pilots follow the ROE, and the ACE must interact with the AWACS crew to identify reported unidentified aircraft (see Figure 7.7 and Section 6.2).

Controls: The controls include the ROE to slow down the engagement and a chain of command to prevent individual error or erratic behavior.

Roles and Responsibilities: The ACE was responsible for controlling combat operations and for ensuring that the ROE were enforced. He flew in the AWACS so he could get up-to-the-minute information about the state of the TAOR airspace.

The ACE was always a highly experienced person with fighter experience. That day, the ACE was a major with 19 years in the Air Force. He had perhaps more combat experience than anyone else in the Air Force under 40. He had logged 2000 total hours of flight time and flown 125 combat missions, including 27 in the Gulf War, during which time he earned the Distinguished Flying Cross and two air medals for heroism. At the time of the accident, he had worked for 4 months as an ACE and flown approximately 15 to 20 missions on the AWACS [108].

The Mission Director on the ground provided a chain of command for real-time decision making from the pilots to the CFAC commander. On the day of the accident, the Mission Director was a lieutenant colonel with over 18 years in the AF. He had logged over 1000 hours in the F-4 in Europe and an additional 100 hours worldwide in the F-15 [108].

Environmental and Behavior-Shaping Factors: No pertinent factors were identified in the reports and books on the accident.

Dysfunctional Interactions at this Level

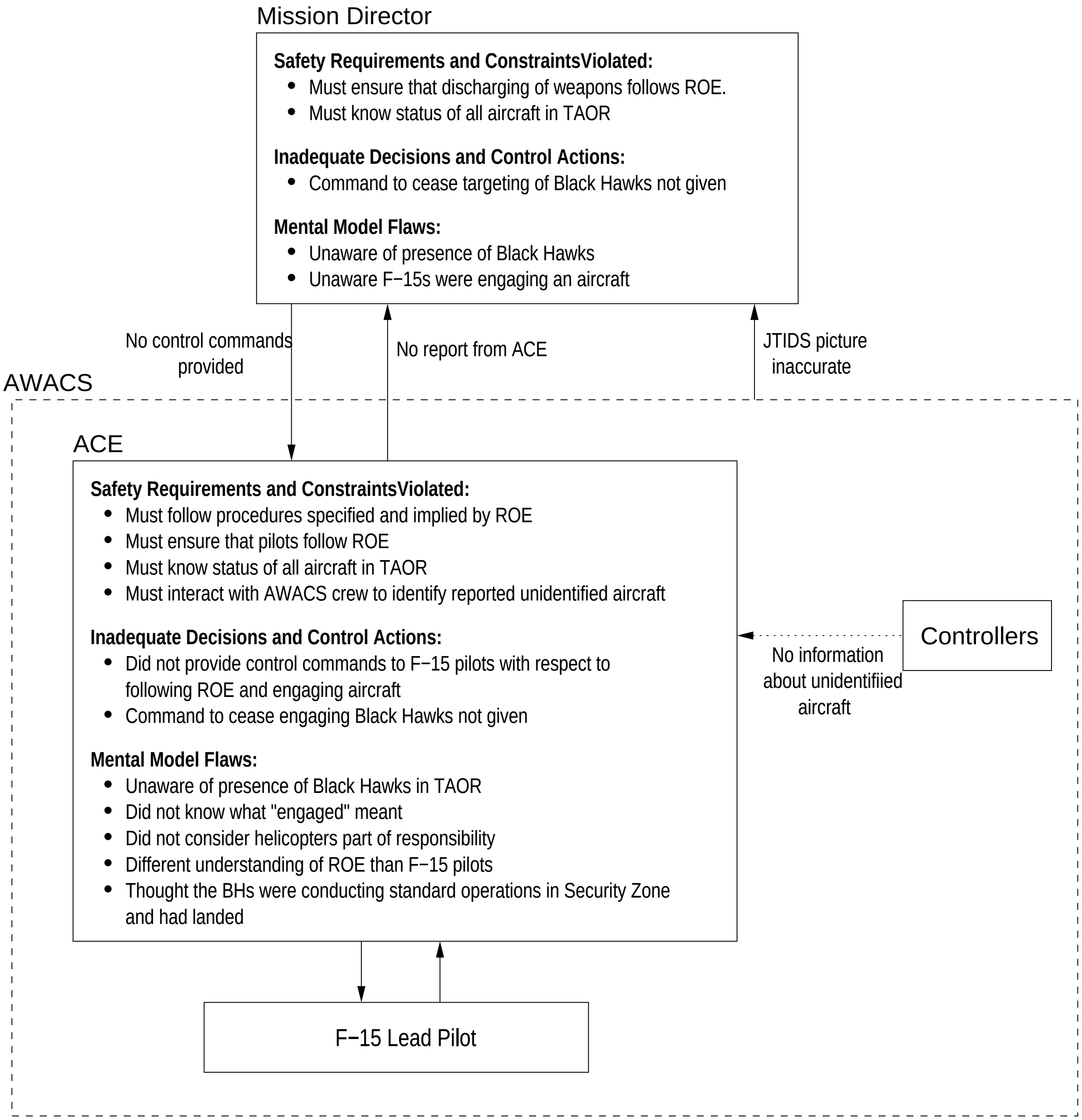
The ACE was supposed to get information about unidentified or enemy aircraft from the AWACS mission crew, but in this instance they did not provide it.

Flawed or Inadequate Decisions and Control Actions

The ACE did not provide any control commands to the F-15s with respect to following the ROE or engaging and firing on the U.S. helicopters.

Reasons for Flawed Control Actions and Dysfunctional Interactions

Incorrect Control Algorithms: The control algorithms should theoretically been effective, but they were never executed.



Mission Director

Safety Requirements and ConstraintsViolated:

- Must ensure that discharging of weapons follows ROE.
- Must know status of all aircraft in TAOR

Inadequate Decisions and Control Actions:

- Command to cease targeting of Black Hawks not given

Mental Model Flaws:

- Unaware of presence of Black Hawks
- Unaware F−15s were engaging an aircraft

ACE

Safety Requirements and ConstraintsViolated:

- Must follow procedures specified and implied by ROE
- Must ensure that pilots follow ROE
- Must know status of all aircraft in TAOR
- Must interact with AWACS crew to identify reported unidentified aircraft

Inadequate Decisions and Control Actions:

- Did not provide control commands to F−15 pilots with respect to following ROE and engaging aircraft
- Command to cease engaging Black Hawks not given

Mental Model Flaws:

- Unaware of presence of Black Hawks in TAOR
- Did not know what "engaged" meant
- Did not consider helicopters part of responsibility
- Different understanding of ROE than F−15 pilots
- Thought the BHs were conducting standard operations in Security Zone and had landed

Figure 7.7: Analysis for the ACE and Mission Director

Inaccurate Mental Models: CFAC, and thus the Mission Director and ACE, exercised ultimate tactical control of the helicopters, but they shared the common view with the AWACS crew that helicopter activities were not an integral part of OPC air operations. In testimony after the accident, the ACE commented "The way I understand it, only as a courtesy does the AWACS track Eagle Flight."

The Mission Director and ACE also did not have the information necessary to exercise their responsibility. The ACE had an inaccurate model of where the Black Hawks were located in the airspace. He testified that he presumed the Black Hawks were conducting standard operations in the Security Zone and had landed [90]. He also testified that, although he had a radarscope, he had no knowledge of AWACS radar symbology: "I have no idea what those little blips mean." The Mission Director, on the ground, was dependent on the information about the current airspace state sent down from the AWACS via JTIDS (the Joint Tactical Information Distribution System).

The ACE testified that he assumed the F-15 pilots would ask him for guidance in any situation involving a potentially hostile aircraft, as required by the ROE. The ACE's and F-15 pilots' mental models of the ROE clearly did not match with respect to who had the authority to initiate the engagement of unidentified aircraft. The rules of engagement stated that the ACE was responsible, but some pilots believed they had authority when an imminent threat was involved. Because of security concerns, the actual ROE used were not disclosed during the accident investigation, but, as argued earlier, the slow, low-flying Black Hawks posed no serious threat to an F-15.

Although the F-15 pilot never contacted the ACE about the engagement, the ACE did hear the call of the F-15 lead pilot to the TAOR controller. The ACE testified to the Accident Investigation Board that he did not intervene because he believed the F-15 pilots were not committed to anything at the visual identification point, and he had no idea they were going to react so quickly. Since being assigned to OPC, he said the procedure had been that when the F-15s or whatever fighter was investigating aircraft, they would ask for feedback from the ACE. The ACE and AWACS crew would then try to rummage around and find out whose aircraft it was and identify it specifically. If they were unsuccessful, the ACE would then ask the pilots for a visual identification [90]. Thus, the ACE probably assumed that the F-15 pilots would not fire at the helicopters without reporting to him first, which they had not done yet. At this point, they had simply requested an identification by the AWACS traffic controller. According to his understanding of the ROE, the F-15 pilots would not fire without his approval unless there was an immediate threat, which there was not. The ACE testified that he expected to be queried by the F-15 pilots as to what their course of action should be.

The ACE also testified at one of the hearings:

> "I really did not know what the radio call 'engaged' meant until this morning. I did not think the pilots were going to pull the trigger and kill those guys. As a previous right seater in an F-111, I thought 'engaged' meant the pilots were going down to do a visual intercept." (Piper)

Coordination among Multiple Controllers: Not applicable

Feedback from Controlled Process: The F-15 lead pilot did not follow the ROE and report the identified aircraft to the ACE and ask for guidance, although the ACE did learn about it from the questions the F-15 pilots posed to the controllers on the AWACS aircraft. The Mission Director got incorrect feedback about the state of the airspace from JTIDS.

Time Lags: An unusual time lag occurred where the lag was in the controller and not in one of the other parts of the control loop.[8] The F-15 pilots responded faster than the ACE (in the AWACS) and Mission Director (on the ground) could issue appropriate control instructions (as required by the ROE) with regard to the engagement.

Changes After the Accident

There were no changes after the accident, although roles were clarified.

7.3.5 AWACS

This level of the control structure contains more examples of inconsistent mental models and asynchronous evolution. In addition, this control level provides interesting examples of the adaptation over time of specified procedures to accepted practice and of coordination problems. There were multiple controllers with confused and overlapping responsibilities for enforcing different aspects of the safety requirements and constraints. The overlaps and boundary areas in the controlled processes led to serious coordination problems among those responsible for controlling aircraft in the TAOR.

Context in which Decisions and Actions Took Place

Safety Requirements and Constraints: The general safety constraint involved in the accident at this level was to prevent misidentification of aircraft by the pilots and any friendly fire that might result. More specific requirements and constraints are shown in Figure 7.8.

Controls: Controls included procedures for identifying and tracking aircraft, training (including simulator missions), briefings, staff controllers, and communication channels. The Senior Director and surveillance officer (ASO) provided real-time oversight of the crew's activities while the Mission Crew Commander (MCC) coordinated all the activities aboard the AWACS aircraft.

The Delta Point system, used since the inception of OPC, provided standard code names for real locations. These code names were used to prevent the enemy, who might be listening to radio transmissions, from knowing the helicopters' flight plans.

Roles and Responsibilities: The AWACS crew were responsible for identifying, tracking, and controlling all aircraft enroute to and from the TAOR; for coordinating air refueling; for providing airborne threat warning and control in the TAOR; and for providing surveillance, detection and identification of all unknown aircraft. Individual responsibilities are described in Section 6.2.

The Staff Weapons Director (instructor) was permanently assigned to Incirlik. He did all incoming briefings for new AWACS crews rotating into Incirlik and accompanied them on their first mission in the TAOR. The OPC leadership recognized the potential for some distance to develop

[8] A similar type of time lag led to the loss of an F-18 when a mechanical failure resulted in inputs arriving at the computer interface faster than the computer was able to process them.

between stateside spin-up training and continuously evolving practice in the TAOR. Therefore, permanent *staff* or *instructor* personnel flew with each new AWACS crew on their maiden flight in Turkey. Two of these staff controllers were on the AWACS the day of the accident to answer any questions that the new crew might have about local procedures and to alert them as to how things were *really* done in OPC—i.e., to handle adaptation of accepted practice from specified procedures.

The Senior Director (SD) had worked as an AWACS controller for 5 years. This was his fourth deployment to OPC, his second as an SD, and his sixtieth mission over the Iraqi TAOR [90]. He worked as a SD over 200 days a year and had logged over 2383 hours flying time [108].

The Enroute Controller, who was responsible for aircraft outside the TAOR, was a first lieutenant with four years in the AF. He had finished AWACS training two years earlier (May 1992) and had served in the Iraqi TAOR previously [108].

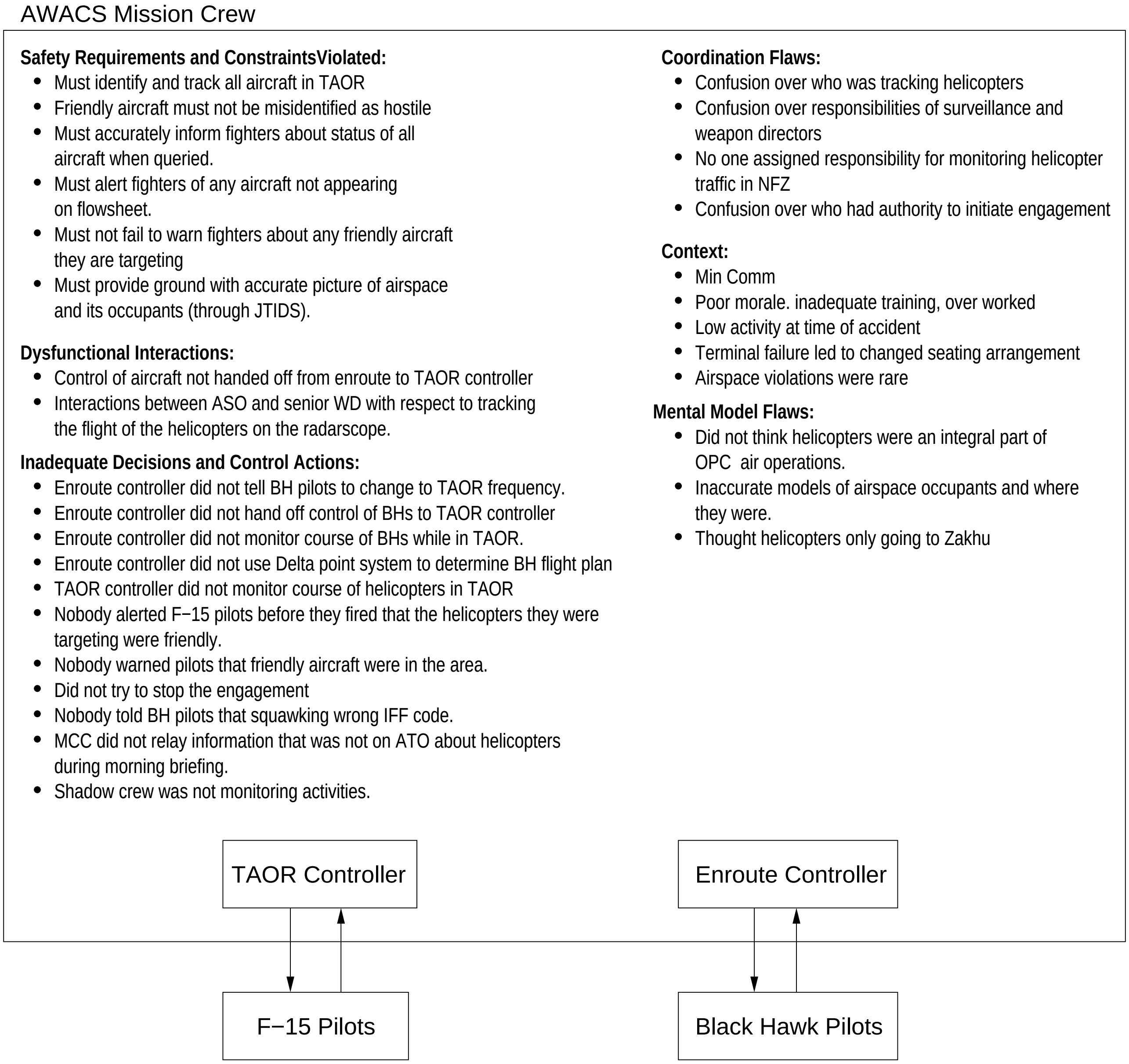
The TAOR Controller, who was responsible for controlling all air traffic flying within the TAOR, was a second lieutenant with over nine years of service in the AF, but he had just finished controller's school and had had no previous deployments outside the continental U.S. In fact, he had become mission ready only two months prior to the incident. This tour was his first in OPC and his first time as a TAOR controller. He had only controlled as a mission-ready weapons director on three previous training flights [108] and never in the role of TAOR controller. AWACS guidance at the time suggested that the most inexperienced controller be placed in the TAOR position: None of the reports on the accident provided the reasoning behind this practice.

The Air Surveillance Officer (ASO) was a captain at the time of the shootdown. She had been mission ready since Oct. 1992 and was rated as an instructor ASO. Because the crew's originally assigned ASO was upgrading and could not make it to Turkey on time, she volunteered to fill in for him. She had already served for five and a half weeks in OPC at the time of the accident and was completing her third assignment to OPC. She worked as an ASO approximately 200 days a year [108].

Environmental and Behavior-Shaping Factors: At the time of the shootdown, shrinking defense budgets were leading to base closings and cuts in the size of the military. At the same time, a changing political climate, brought about by the fall of the Soviet Union, demanded significant U.S. military involvement in a series of operations. The military (including the AWACS crews) were working at a greater pace than they had ever experienced due to budget cuts, early retirements, force outs, slowed promotions, deferred maintenance, and delayed fielding of new equipment. All of these factors contributed to poor morale, inadequate training, and high personnel turnover.

AWACS crews are stationed and trained at Tinker Air Force Base in Oklahoma and then deployed to locations around the world for rotations lasting approximately 30 days. Although all but one of the AWACS controllers on the day of the accident had served previously in the Iraqi no-fly-zone, this was their first day working together and, except for the surveillance officer, the first day of their current rotation. Due to last minute orders, the team got only minimal training, including one simulator session instead of the two full 3-hour sessions required prior to deploying. In the only session they did have, some of the members of the team were missing—the ASO, ACE, and MCC were unable to attend—and one was later replaced: As noted, the ASO originally designated and trained to deploy with this crew was instead shipped off to a career school at the last minute, and another ASO, who was just completing a rotation in Turkey, filled in.

The one simulator session they did receive was less than effective, partly because the computer tape provided by Boeing to drive the exercise was not current (another instance of asynchronous



AWACS Mission Crew

Safety Requirements and ConstraintsViolated:

- Must identify and track all aircraft in TAOR
- Friendly aircraft must not be misidentified as hostile
- Must accurately inform fighters about status of all aircraft when queried.
- Must alert fighters of any aircraft not appearing on flowsheet.
- Must not fail to warn fighters about any friendly aircraft they are targeting
- Must provide ground with accurate picture of airspace and its occupants (through JTIDS).

Dysfunctional Interactions:

- Control of aircraft not handed off from enroute to TAOR controller
- Interactions between ASO and senior WD with respect to tracking the flight of the helicopters on the radarscope.

Inadequate Decisions and Control Actions:

- Enroute controller did not tell BH pilots to change to TAOR frequency.
- Enroute controller did not hand off control of BHs to TAOR controller
- Enroute controller did not monitor course of BHs while in TAOR.
- Enroute controller did not use Delta point system to determine BH flight plan
- TAOR controller did not monitor course of helicopters in TAOR
- Nobody alerted F−15 pilots before they fired that the helicopters they were targeting were friendly.
- Nobody warned pilots that friendly aircraft were in the area.
- Did not try to stop the engagement
- Nobody told BH pilots that squawking wrong IFF code.
- MCC did not relay information that was not on ATO about helicopters during morning briefing.
- Shadow crew was not monitoring activities.

Coordination Flaws:

- Confusion over who was tracking helicopters
- Confusion over responsibilities of surveillance and weapon directors
- No one assigned responsibility for monitoring helicopter traffic in NFZ
- Confusion over who had authority to initiate engagement

Context:

- Min Comm
- Poor morale. inadequate training, over worked
- Low activity at time of accident
- Terminal failure led to changed seating arrangement
- Airspace violations were rare

Mental Model Flaws:

- Did not think helicopters were an integral part of OPC air operations.
- Inaccurate models of airspace occupants and where they were.
- Thought helicopters only going to Zakhu

Figure 7.8: The Analysis at the AWACS control level.

evolution). For example, the maps were out of date and the Rules of Engagement used were different and much more restrictive than those currently in force in OPC. No Mode I codes were listed. The list of friendly participants in OPC did not include UH-60s (Black Hawks) and so on. The second simulation session was canceled because of a wing exercise.

Because the TAOR area had not yet been sanitized, it was a period of low activity: At the time, there were still only four aircraft over the No-Fly-Zone—the two F-15s and the two Black Hawks. AWACS crews are trained and equipped to track literally hundreds of enemy and friendly aircraft during a high-intensity conflict. Many accidents occur during periods of low activity when vigilance is reduced compared to periods of higher activity.

The MCC sits with the other two key supervisors (SD and ACE) toward the front of the aircraft in a three-seat arrangement named the "Pit," where each has his own radarscope. The SD is seated to the MCC's left. Surveillance is seated in the rear. Violations of the No-Fly-Zone had been rare and threats few during the past three years, so that day's flight was expected to be an average one and the supervisors in the Pit anticipated just another routine mission [90]

During the initial orbit of the AWACS, the technicians determined that one of the radar consoles was not operating. According to Snook, this type of problem was not uncommon, and the AWACS is therefore designed with extra crew positions. When the enroute controller realized his assigned console was not working properly, he moved from his normal position between the TAOR and tanker controllers, to a spare seat directly behind the Senior Director. This position kept him out of the view of his supervisor and also eliminated physical contact with the TAOR controller.

Dysfunctional Interactions Among the Controllers

According to the formal procedures, control of aircraft was supposed to be handed off from the enroute controller to the TAOR controller when the aircraft entered the TAOR. This handoff did not occur for the Black Hawks, and the TAOR controller was not made aware of the Black Hawks' flight within the TAOR. Snook explains this communication error as resulting from the terminal failure, which interfered with communication between the TAOR and enroute controllers. But this explanation does not gibe with the fact that the *normal* procedure of the enroute controller was to continue to control helicopters without handing them off to the TAOR controller, even when the enroute and TAOR controllers were seated in their usual places next to each other. There may usually have been more informal interaction about aircraft in the area when they were seated next to each other, but there is no guarantee that such interaction would have occurred even with a different seating arrangement. Note that the helicopters had been dropped from the radar screens and the enroute controller had an incorrect mental model of where they were: He thought they were close to the boundary of the TAOR and was unaware they had gone deep within it. The enroute controller, therefore, could not have told the TAOR controller about the true location of the Blackhawks even if they had been sitting next to each other.

The interaction between the surveillance officer and the senior weapons director with respect to tracking the helicopter flight on the radar screen involved many dysfunctional interactions. For example, the surveillance officer put an attention arrow on the senior director's radar scope in an attempt to query him about the lost helicopter symbol that was floating, at one point, unattached to any track. The senior director did not respond to the attention arrow, and it automatically dropped off the screen after 60 seconds. The helicopter symbol (H) dropped off the radar screen when the radar and IFF returns from the Black Hawks faded and did not return until just before the engagement, removing any visual reminder to the AWACS crew that there were Black Hawks inside the TAOR. The accident investigation did not include an analysis of the design of the AWACS human-computer interface nor how it might have contributed to the accident.

During his court martial for negligent homicide, the senior director argued that his radarscope did not identify the helicopters as friendly and that therefore he was not responsible. When asked why the Black Hawk identification was dropped from the radar scope, he gave two reasons. First, because it was no longer attached to any active signal, they assumed the helicopter had landed somewhere. Second, because the symbol displayed on their scopes was being relayed in real time through a JTIDS downlink to commanders on the ground, they were very concerned about sending out an inaccurate picture of the TAOR.

> "Even if we suspended it, it would not be an accurate picture, because we wouldn't know for sure if that is where he landed. Or if he landed several minutes earlier, and where that would be. So, the most accurate thing for us to do at that time, was to drop the symbology [*sic*]."

Flawed or Inadequate Decision Making and Control Actions

There were myriad inadequate control actions in this accident, involving each of the controllers in the AWACS. The AWACS crew work as a team so it is sometimes hard to trace incorrect decisions to one individual. While from each individual's standpoint the actions and decisions may have been correct, when put together as a whole the decisions were incorrect.

The enroute controller never told the Black Hawk pilots to change to the TAOR frequency that was being monitored by the TAOR controller and did not hand off control of the Black Hawks to the TAOR controller. The established practice of not handing off the helicopters had probably evolved over time as a more efficient way of handling traffic. Because the helicopters were usually only at the very border of the TAOR and spent very little time there, the overhead of handing them off twice within a short time period was considered inefficient by the AWACS crews. As a result, the procedures used had changed over time to the more efficient procedure of keeping them under the control of the enroute controller. The AWACS crews were not provided with written guidance or training regarding the control of helicopters within the TAOR, and, in its absence, they adapted their normal practices for fixed-wing aircraft as best they could to apply them to helicopters.

In addition to not handing off the helicopters, the enroute controller did not monitor the course of the Black Hawks while they were in the TAOR (after leaving Zakhu), did not take note of the flight plan (from *Whiskey* to *Lima*), did not alert the F-15 pilots there were friendly helicopters in the area, did not alert the F-15 pilots before they fired that the helicopters they were targeting were friendly, and did not tell the Black Hawk pilots that they were on the wrong frequency and were squawking the wrong IFF Mode I code.

The TAOR controller did not monitor the course of the Black Hawks in the TAOR and did not alert the F-15 pilots before they fired that the helicopters they were targeting were friendly. None of the controllers warned the F-15 pilots at any time that there were friendly helicopters in the area nor did they try to stop the engagement. The accident investigation board found that because Army helicopter activities were not normally known at the time of the fighter pilots' daily briefings, normal procedures were for the AWACS crews to receive real-time information about their activities from the helicopter crews and to relay that information on to the other aircraft in the area. If this truly was established practice, it clearly did not occur on that day.

The controllers were supposed to be tracking the helicopters using the Delta Point system, and the Black Hawk pilots had reported to the enroute controller that they were traveling from *Whiskey* to *Lima*. The enroute controller testified, however, that he had no idea of the towns to which the code names *Whiskey* and *Lima* referred. After the shootdown, he went in search of the card defining the call signs and finally found it in the Surveillance Section [90]. Clearly, tracking helicopters using call signs was not a common practice or the charts would have been closer at hand. In fact, during the court martial of the Senior Director, the defense was unable to locate any AWACS crewmember at Tinker AFB (where AWACS crews were stationed and trained) who could testify that he or she had *ever* used the Delta Point system [90] although clearly the Black Hawk pilots thought it was being used because they provided their flight plan using Delta Points.

None of the controllers in the AWACS told the Black Hawk helicopters that they were squawking the wrong IFF code for the TAOR. Snook cites testimony from the court martial of the Senior Director that posits three related explanations for this lack of warning: (1) the minimum communication policy, (2) a belief by the AWACS crew that the Black Hawks should know what they were doing, and (3) pilots not liking to be told what to do. None of these explanations provided during the trial is very satisfactory and appear to be after-the-fact rationalizations for the controllers not doing their job when faced with possible court martial and jail terms. Given that the controllers acknowledged that the Army helicopters never squawked the right codes and had not for months, there must have been other communication channels that could have been used besides real-time radio communication to remedy this situation so the min comm policy is not an adequate explanation. Arguing that the pilots should know what they were doing is simply an abdication of responsibility, as is the argument that pilots did not like being told what to do. A different perspective, and one that likely applies to all the controllers, was provided by the Staff Weapons Director who testified that "For a helicopter, if he's going to Zakhu, I'm not that concerned about him going beyond that. So, I'm not really concerned about having an F-15 needing to identify this guy."

The Mission Crew Commander had provided the crew's morning briefing. He spent some time going over the activity flow sheet, which listed all the friendly aircraft flying in the OPC that day, their call signs, and the times they were scheduled to enter the TAOR. According to Piper (but nobody else mentions it), he failed to note the helicopters, even though their call signs and their IFF information had been written on the margin of his flowsheet.

The shadow crew always flew with new crews on their first day in OPC, but the task of these instructors does not seem to have been well defined. At the time of the shootdown, one was in the galley "taking a break" and the other went back to the crew rest area, read a book, and took a nap. The staff weapons director, who was asleep in the back of the AWACS, during the court martial of the Senior Director testified that his purpose on the mission was to be the "answer man," just to answer any questions they might have. This was a period of very little activity in the area (only the two F-15s were supposed to be in the TAOR) and the shadow crew members may have thought their advice was not needed at that time.

When the staff weapons director went back to the rest area, the only symbol displayed on the scopes of the AWACS controllers was the one for the helicopters (*EE01*), which they thought were going to Zakhu only.

Because many of the dysfunctional actions of the crew *did* conform to the established practice (e.g., not handing off helicopters to the TAOR controller), it is unclear what different result might have occurred if the shadow crew had been in place. For example, the staff Weapon Director testified during the hearings and trial that he had seen helicopters out in the TAOR before, past Zakhu, but he really did not feel it was necessary to brief crews about the Delta Point system to determine a helicopter's destination [90].[9]

Reasons for the Flawed Control

Inadequate control algorithms: This level of the accident analysis provides an interesting example of the difference between prescribed procedures and established practice, the adaptation of procedures over time, and migration toward the boundaries of safe behavior. Because of the many helicopter missions that ran from Diyarbakir to Zakhu and back, the controllers testified that it did not seem worth handing them off and switching them over to the TAOR frequency for only a few minutes. Established practice (keeping the helicopters under the control of the enroute controller instead of handing them off to the TAOR controller) appeared to be safe until the day the helicopters' behavior differed from normal, i.e., they stayed longer in the TAOR and ventured beyond a few miles inside the boundaries. The effective procedures no longer assured safety under these conditions. A complicating factor in the accident was the universal misunderstanding of each of the controllers' responsibilities with respect to tracking Army helicopters.

Snook suggests that the *min comm* norm contributed to the AWACS crew's general reluctance to enforce rules, contributed to AWACS not correcting Eagle Flight's improper Mode I code, and discouraged controllers from pushing helicopter pilots to the TAOR frequency when they entered Iraq because they were reluctant to say more than absolutely necessary.

According to Snook, there were also no explicit or written procedures regarding the control of helicopters. He states that radio contact with helicopters was lost frequently, but there were no procedures to follow when this occurred. In contrast, Piper claims the AWACS operations manual says:

> "Helicopters are a high interest track and should be hard copied every five minutes in turkey and every two minutes in Iraq. These coordinates should be recorded in a special log book, because radar contact with helicopters is lost and the radar symbology [*sic*]can be suspended"[90].

There is no information in the publicly available parts of the accident report about any special logbook or whether such a procedure was normally followed.

Inaccurate and Inconsistent Mental Models: In general, the AWACS crew (and the ACE) shared the common view that helicopter activities were not an integral part of OPC air operations. There was also a misunderstanding about which provisions of the ATO applied to Army helicopter activities.

Most of the people involved in the control of the F-15s were unaware of the presence of the Black Hawks in the TAOR that day, the lone exception perhaps being the enroute controller who knew they were there but apparently thought they would stay at the boundaries of the TAOR and thus were far from their actual location deep within it. The TAOR controller testified that he had never talked to the Black Hawks: Following their two check-ins with the enroute controller, the

[9] Even if the actions of the shadow crew did not contribute to this particular accident, we can take advantage of the accident investigation to perform a safety audit on the operation of the system and identify potential improvements.

helicopters had remained on the enroute frequency (as was the usual, accepted practice), even as they flew deep into the TAOR.

The enroute controller, who had been in contact with the Black Hawks, had an inaccurate model of where the helicopters were. When the Black Hawk pilots originally reported their takeoff from the Army Military Coordination Center at Zakhu, they contacted the enroute controller and said they were bound for *Lima*. The enroute controller did not know to what city the call sign *Lima* referred and did not try to look up this information. Other members of the crew also had inaccurate models of their responsibilities, as described in the next section. The Black Hawk pilots clearly thought the AWACS was tracking them and also thought the controllers were using the Delta Point system (or the pilots would not have provided the route names in that way).

The AWACS crews did not appear to have accurate models of the Black Hawks mission and role in OPC. Some of the flawed control actions seem to have resulted from a mental model that helicopters only went to Zakhu and therefore did not need to be tracked or to follow the standard TAOR procedures.

As with the pilots and their visual recognition training, the incorrect mental models may have been at least partially the result of the inadequate AWACS training the team received.

Coordination Among Multiple Controllers: As mentioned earlier, coordination problems are pervasive in this accident due to overlapping control responsibilities and confusion about responsibilities in the boundary areas of the controlled process. Most notably, the helicopters usually operated close to the boundary of the TAOR, resulting in confusion over who was or should be controlling them.

The official Accident Report noted a significant amount of confusion within the AWACS mission crew regarding the tracking responsibilities for helicopters [4]. The Mission Crew Commander testified that nobody was specifically assigned responsibility for monitoring helicopter traffic in the No-Fly-Zone and that his crew believed the helicopters were not included in their orders [90]. The Staff Weapons Director made a point of not knowing what the Black Hawks do: “It was some kind of a squirrely mission” [90]. During the court martial of the Senior Director, the AWACS tanker controller testified that in the briefing the crew received upon arrival at Incirlik, the Staff Weapons Director had said about helicopters flying in the No-Fly-Zone, ”They’re there, but don’t pay any attention to them.” The Enroute controller testified that the handoff procedures applied only to fighters. “We generally have no set procedures for any of the helicopters. . . . We never had any [verbal] guidance [or training] at all on helicopters.

Coordination problems also existed between the activities of the surveillance personnel and the other controllers. During the investigation of the accident, the ASO testified that surveillance’s responsibility was south of the 36th parallel and the other controllers were responsible for tracking and identifying all aircraft north of the 36th parallel. The other controllers suggested that surveillance was responsible for tracking and identifying all unknown aircraft, regardless of location. In fact, Air Force regulations say that surveillance had tracking responsibility for unknown and unidentified tracks throughout the TAOR. It is not possible through the testimony alone, again because of the threat of court martial, to piece out exactly what was the problem here, including simply a migration of normal operations from specified operations. At the least, it is clear that there was confusion about who was in control of what.

One possible explanation for the lack of coordination among controllers at this level of the hierarchical control structure is that, as suggested by Snook, this particular group had never trained as a team. But given the lack of procedures for handling helicopters and the confusion even by experienced controllers and the staff instructors about responsibilities for handling helicopters, Snook's explanation is not very convincing. A more plausible explanation is simply a lack of guidance and delineation of responsibilities by the management level above. And even if the roles of everyone in such a structure had been well-defined originally, uncontrolled local adaptation to more efficient procedures and asynchronous evolution of the different parts of the control structure created dysfunctionalities as time passed. The helicopters and fixed wing aircraft had separate control structures that only joined fairly high up on the hierarchy and, as is described in the next section, there were communication problems between the components at the higher levels of the control hierarchy, particularly between the Army Military Coordination Center (MCC) and the Combined Forces Air Component (CFAC) headquarters.

Feedback from the Controlled Process: Signals to the AWACS from the Black Hawks were inconsistent due to line-of-sight limitations and the mountainous terrain in which the Black Hawks were flying. The helicopters used the terrain to mask themselves from air defense radars, but this terrain masking also caused the radar returns from the Black Hawks to the AWACS (and to the fighters) to fade at various times.

Time Lags: Important time lags contributed to the accident, such as the delay of radio reports from the Black Hawk helicopters due to radio signal transmission problems and their inability to use the TACSAT radios until they had landed. As with the ACE, the speed with which the F-15 pilots acted also provided the controllers with little time to evaluate the situation and respond appropriately.

Changes After the Accident

Many changes were instituted with respect to AWACS operations after the accident:

- Confirmation of a positive IFF Mode IV check was required for all OPC aircraft prior to their entry into the TAOR.
- The responsibilities for coordination of air operations were better defined.
- All AWACS aircrews went through a one-time retraining and recertification program, and every AWACS crewmember had to be recertified.
- A plan was produced to reduce the temporary duty of AWACS crews to 120 days a year. In the end, it was decreased from 166 to 135 days per year from January 1995 to July 1995. The Air Combat Command planned to increase the number of AWACS crews.
- AWACS control was required for all TAOR flights.
- In addition to normal responsibilities, AWACS controllers were required to specifically maintain radar surveillance of all TAOR airspace and to issue advisory/deconflicting assistance on all operations, including helicopters.
- The AWACS controllers were required to periodically broadcast friendly helicopter locations operating in the TAOR to all aircraft.

Although not mentioned anywhere in the available documentation on the accident, I assume also that either the AWACS crews started to use the Delta Point system or the Black Hawk pilots were told not to use it and an alternative means for transmitting flight plans was mandated.

7.3.6 Military Coordination Center, Combined Forces Air Component, and CTF Commander

Fully understanding the behavior at any level of the socio-technical control structure requires understanding how and why the control at the next higher level allowed or contributed to the inadequate control at the current level. In this accident, many of the erroneous decisions and control actions at the lower levels can only be fully understood by examining this level of control.

Context in which Decisions and Actions Took Place

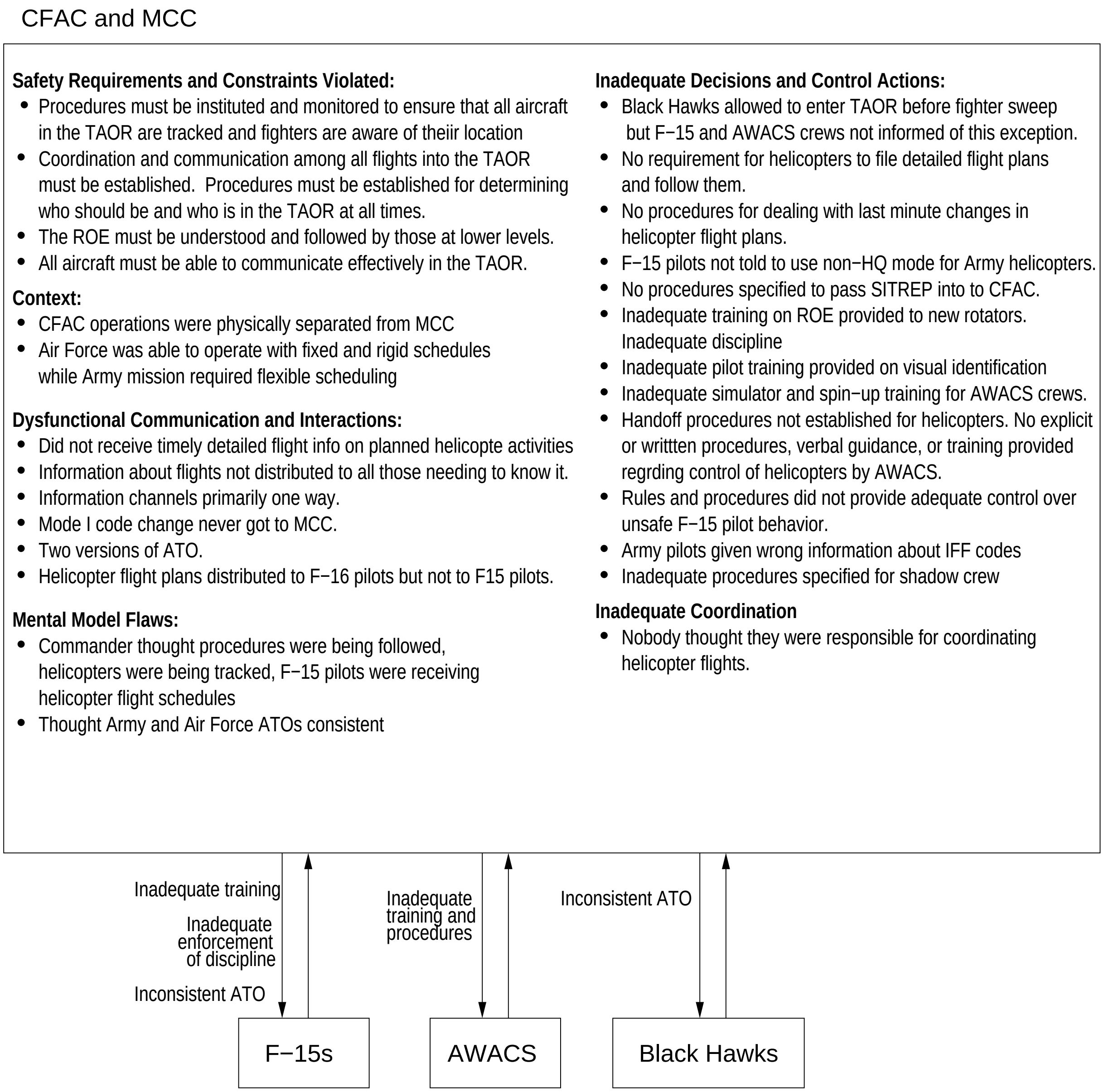
Safety Requirements and Constraints Violated: There were many safety constraints violated at this level of the control structure, and several people were investigated for potential court martial and received official letters of reprimand. These safety constraints include: (1) procedures must be instituted that delegate appropriate responsibility, specify tasks, and provide effective training to all those responsible for tracking aircraft and conducting combat operations; (2) procedures must be consistent or at least complementary for everyone involved in TAOR airspace operations; (3) performance must be monitored (feedback channels established) to ensure that safety-critical activities are being carried out correctly and that local adaptations have not moved operations beyond safe limits; (4) equipment and procedures must be coordinated between the Air Force and Army to make sure that communication channels are effective and that asynchronous evolution has not occurred; (5) accurate information about scheduled flights must be provided to the pilots and the AWACS crews.

Controls: The controls in place included operational orders and plans to designate roles and responsibilities as well as a management structure, the ACO, coordination meetings and briefings, a chain of command (OPC commander to Mission Director to ACE to Pilots), disciplinary actions for those not following the written rules, and a group (the Joint Operations and Intelligence Center or JOIC) responsible for ensuring effective communication occurred.

Roles and Responsibilities: The MCC had operational control over the Army helicopters while the CFAC had operational control over fixed-wing aircraft and tactical control over *all* aircraft in the TAOR. The Combined Task Force Commander General (who was above both the CFAC and MCC) had ultimate responsibility for the coordination of fixed-wing aircraft flights with Army helicopters.

While specific responsibilities of individuals might be considered here in an official accident analysis, treating the CFAC and MCC as entities is sufficient for the purposes of this example analysis.

Environmental and Behavior-Shaping Factors: The Air Force operated on a predictable, well-planned, and tightly executed schedule. Detailed mission packages were organized weeks and



CFAC and MCC

Safety Requirements and Constraints Violated:

- Procedures must be instituted and monitored to ensure that all aircraft in the TAOR are tracked and fighters are aware of theiir location
- Coordination and communication among all flights into the TAOR must be established. Procedures must be established for determining who should be and who is in the TAOR at all times.
- The ROE must be understood and followed by those at lower levels.
- All aircraft must be able to communicate effectively in the TAOR.

Context:

- CFAC operations were physically separated from MCC
- Air Force was able to operate with fixed and rigid schedules while Army mission required flexible scheduling

Dysfunctional Communication and Interactions:

- Did not receive timely detailed flight info on planned helicopte activities
- Information about flights not distributed to all those needing to know it.
- Information channels primarily one way.
- Mode I code change never got to MCC.
- Two versions of ATO.
- Helicopter flight plans distributed to F−16 pilots but not to F15 pilots.

Mental Model Flaws:

- Commander thought procedures were being followed, helicopters were being tracked, F−15 pilots were receiving helicopter flight schedules
- Thought Army and Air Force ATOs consistent

Inadequate Decisions and Control Actions:

- Black Hawks allowed to enter TAOR before fighter sweep but F−15 and AWACS crews not informed of this exception.
- No requirement for helicopters to file detailed flight plans and follow them.
- No procedures for dealing with last minute changes in helicopter flight plans.
- F−15 pilots not told to use non−HQ mode for Army helicopters.
- No procedures specified to pass SITREP into to CFAC.
- Inadequate training on ROE provided to new rotators. Inadequate discipline
- Inadequate pilot training provided on visual identification
- Inadequate simulator and spin−up training for AWACS crews.
- Handoff procedures not established for helicopters. No explicit or writtten procedures, verbal guidance, or training provided regrding control of helicopters by AWACS.
- Rules and procedures did not provide adequate control over unsafe F−15 pilot behavior.
- Army pilots given wrong information about IFF codes
- Inadequate procedures specified for shadow crew

Inadequate Coordination

- Nobody thought they were responsible for coordinating helicopter flights.

Figure 7.9: Analysis at the CFAC-MCC level.

months in advance. Rigid schedules were published and executed in preplanned *packages.* In contrast, Army aviators had to react to constantly changing local demands, and they prided themselves on their flexibility [108]. Because of the nature of their missions, exact takeoff times and detailed flight plans for helicopters were virtually impossible to schedule in advance. They were even more difficult to execute with much rigor. The Black Hawks' flight plan contained their scheduled takeoff time, transit routes between Diyarbakir through Gate 1 to Zakhu, and return time. Because the Army helicopter crews rarely knew exactly where they would be going within the TAOR until after they were briefed at the Military Coordination Center at Zakhu, most flight plans only indicated that Eagle Flight would be "operating in and around the TAOR."

The physical separation of the Army Eagle Flight pilots from the CFAC operations and Air Force pilots at Incirlik contributed to the communication difficulties that already existed between the services.

Dysfunctional Interactions Among Controllers

Dysfunctional communication at this level of the control structure played a critical role in the accident. These communication flaws contributed to the coordination flaws at this level and at the lower levels.

A critical safety constraint to prevent friendly fire required that the pilots of the fighter aircraft know who was in the No-Fly-Zone and whether they were supposed to be there. However, neither the CTF staff nor the Combined Forces Air Component staff requested nor received timely, detailed flight information on planned MCC helicopter activities in the TAOR. Consequently, the OPC daily Air Tasking Order was published with little detailed information regarding U.S. helicopter flight activities over northern Iraq.

According to the official accident report, specific information on routes of flight and times of MCC helicopter activity in the TAOR was normally available to the other OPC participants only when AWACS received it from the helicopter crews by radio and relayed the information on to the pilots [4]. While those at the higher levels of control may have thought this relaying of flight information was occurring, that does not seem to be the case given that the Delta point system (wherein the helicopter crews provided the AWACS controllers with their flight plan) was not used by the AWACS controllers: When the helicopters went beyond Zakhu, the AWACS controllers did not know their flight plans and therefore could not relay that information to the fighter pilots and other OPC participants.

The weekly flight schedules the MCC provided to the CFAC staff were not complete enough for planning purposes. While the Air Force could plan their missions in advance, the different type of Army helicopter missions had to be flexible to react to daily needs. The MCC daily mission requirements were generally based on the events of the previous day. A weekly flight schedule was developed and provided to the CTF staff, but a firm itinerary was usually not available until after the next day's ATO was published. The weekly schedule was briefed at the CTF staff meetings on Mondays, Wednesday, and Fridays, but the information was neither detailed nor firm enough for effective rotary-wing and fixed-wing aircraft coordination and scheduling purposes [4].

Each daily ATO was published showing several Black Hawk helicopter lines. Of these, two helicopter lines (two flights of two helicopters each) were listed with call signs (Eagle 01/02 and Eagle 03/04), mission numbers, IFF Mode II codes, and a route of flight described only as LLTC (the identifier for Diyarbakir) to TAOR to LLTC. No information regarding route or duration of flight time within the TAOR was given on the ATO. Information concerning takeoff time and entry time into the TAOR was listed as *A/R* (as required).

Every evening, the MCC at Zakhu provided a Situation Report (SITREP) to the JOIC (located at Incirlik), listing the helicopter flights for the following day. The SITREP did not contain complete flight details and arrived too late to be included in the next day's ATO. The MCC would call the JOIC the night prior to the scheduled mission to "activate" the ATO line. There were, however, no procedures in place to get the SITREP information from the JOIC to those needing to know it in CFAC.

After receiving the SITREP, a duty officer in the JOIC would send takeoff times and gate times (the times the helicopters would enter Northern Iraq) to Turkish operations for approval. Meanwhile, an intelligence representative to the JOIC consolidated the MCC weekly schedule with the SITREP and used secure intelligence channels to pass this updated information to some of his counterparts in operational squadrons who had requested it. No procedures existed to pass this information from the JOIC to those in CFAC with tactical responsibility for the helicopters (through the ACE and Mission Director) [4]. Because CFAC normally determined who would fly when, the information channels were designed primarily for one-way communications outward and downward.

In the specific instance involved in the shootdown, the MCC weekly schedule was provided on April 8 to the JOIC and thence to the appropriate person in CFAC. That schedule showed a two-ship, MCC helicopter administrative flight scheduled for April 14. According to the official accident report, two days before (April 12) the MCC Commander had requested approval for an April 14 flight outside the Security Zone from Zakhu to the towns of Irbil and Salah ad Din. The OPC Commanding General approved the written request on April 13, and the JOIC transmitted the approval to the MCC but apparently the information was not provided to those responsible for producing the ATO. The April 13 SITREP from MCC listed the flight as "mission support," but contained no other details. Note more information was available earlier than normal in this instance, and it could have been included in the ATO but the established communication channels and procedures did not exist to get it to the right places. The MCC weekly schedule update, received by the JOIC on the evening of April 13 along with the MCC SITREP, gave the destinations for the mission as Salah ad Din and Irbil. This information was not passed to CFAC.

Late in the afternoon on April 13, MCC contacted the JOIC duty officer and activated the ATO line for the mission. A takeoff time of 0520 and a gate time of 0625 were requested. No takeoff time or route of flight beyond Zakhu was specified. The April 13 SITREP, the weekly flying schedule update, and the ATO-line activation request were received by the JOIC too late to be briefed during the Wednesday (April 13) staff meetings. None of the information was passed to the CFAC scheduling shop (which was responsible for distributing last minute changes to the ATO through various sources such as the Battle Staff Directives, morning briefings, etc.), to the ground-based Mission Director, nor to the ACE on board the AWACS [4]. Note that this flight was not a routine food and medical supply run, but instead it carried sixteen high-ranking VIPs and required the personal attention and approval of the CTF Commander. Yet information about the flight was never communicated to the people who needed to know about it [108]. That is, the information went up from the MCC to the CTF staff, but not across from MCC to CFAC nor down from the CTF staff to CFAC (see Figure 6.4).

A second example of a major dysfunctional communication involved the communication of the proper radio frequencies and IFF codes to be used in the TAOR. About two years before the shootdown, someone in the CFAC staff decided to change the instructions pertaining to IFF modes and codes. According to Snook, no one recalled exactly how or why this change occurred. Before the change, all aircraft squawked a single Mode I code everywhere they flew. After the change, all aircraft were required to switch to a different Mode I code while flying in the No-Fly-Zone. The change was communicated through the daily ATO. However, after the accident it was discovered that the Air Force's version of the ATO was not exactly the same as the one received electronically by the Army aviators—another instance of asynchronous evolution and lack of linkup between system components. For at least two years, there existed two versions of the daily ATO: one printed out directly by the Incirlik Frag Shop and distributed locally by messenger to all units at Incirlik Air Base, and a second one transmitted electronically through an Air Force communications center (the JOIC) to Army helicopter operations at Diyarbakir. The one received by the Army aviators was identical in all respects to the one distributed by the Frag Shop, *except* for the changed Mode I code information contained in the SPINS. The ATO that Eagle Flight received contained no mention of two Mode I codes [108].

What about the confusion about the proper radio frequency to be used by the Black Hawks in the TAOR? Piper notes that the Black Hawk pilots were told to use the enroute frequency while flying in the TAOR. The Commander of OPC testified after the accident that the use by the Black Hawks of the enroute radio frequency rather than the TAOR frequency had been briefed to him as a *safety measure* because the Black Hawk helicopters were not equipped with HAVE QUICK technology. The ACO required the F-15s to use non-HAVE QUICK mode when talking to specific types of aircraft (such as F-1s) that, like the Black Hawks, did not have the new technology. The list of non-HQ aircraft provided to the F-15 pilots, however, for some reason did not include UH-60s. Apparently the decision was made to have the Black Hawks use the enroute radio frequency but this decision was never communicated to those responsible for the F-15 procedures specified in the ACO (Aircraft Control Order). Note that a thorough investigation of the higher levels of control is necessary to explain properly the use of the enroute radio frequency by the Black Hawks. Of the various reports on the shootdown, only Piper notes the fact that an exception had been made for Army helicopters for safety reasons—the official accident report, Snook's detailed book on the accident, and the GAO report do not mention this fact! Piper found out about it from her attendance at the public hearings and trial. This omission of important information from the accident reports is an interesting example of how incomplete investigation of the higher levels of control can lead to incorrect causal analysis. In her book, Piper questions why the Accident Investigation Board, while producing 21 volumes of evidence, never asked the Commander of OPC about the radio frequency and other problems found during the investigation.

Other official exceptions were made for the helicopter operations, such as allowing them in the Security Zone without AWACS coverage. Using STAMP, the accident can be understood as a process where the operations of the Army and Air Force adapted and diverged without effective communication and coordination.

Many of the dysfunctional communications and interactions stem from asynchronous evolution of the mission and the operations plan. In response to the evolving mission in Northern Iraq, air assets were increased in September 1991 and a significant portion of the ground forces were withdrawn. Although the original organizational structure of the CTF was modified at this time, the operations plan was not. In particular, the position of the person who was in charge of communication and coordination between the MCC and CFAC was eliminated without establishing an alternative communication channel.

Flawed or Inadequate Control Actions

There were many flawed or missing control actions at this level, including:

- *The Black Hawk pilots were allowed to enter the TAOR without AWACS coverage and the F-15 pilots and AWACS crews were not informed about this exception to the policy.* This control problem is an example of the problems of distributed decision making with other decision makers not being aware of the decisions of others (see the Zeebrugge example in Section 3.3).

  Prior to September 1993, Eagle Flight helicopters flew any time required, prior to fighter sweeps and without fighter coverage, if necessary. After September 1993, helicopter flights were restricted to the security zone if AWACS and fighter coverage were not on station. But for the mission on April 14, Eagle Flight requested and received permission to execute their flight outside the security zone. A CTF policy letter dated September 1993 implemented the following policy for UH-60 helicopter flights supporting the MCC: "All UH-60 flights into Iraq outside of the security zone require AWACS coverage." Helicopter flights had routinely been flown within the TAOR security zone without AWACS or fighter coverage and CTF personnel at various levels were aware of this. MCC personnel were aware of the requirement to have AWACS coverage for flights outside the security zone and complied with that requirement. However, the F-15 pilots involved in the accident, relying on the written guidance in the ACO, believed that no OPC aircraft, fixed or rotary wing, were allowed to enter the TAOR prior to a fighter sweep [4].

  At the same time, the Black Hawks also thought they were operating correctly. The Army Commander at Zakhu had called the Commander of Operations, Plans, and Policy for OPC the night before the shootdown and asked to be able to fly the mission without AWACS coverage. He was told that they must have AWACS coverage. From the view of the Black Hawks pilots (who had reported in to the AWACS during the flight and provided their flight plan and destinations) they were complying and were under AWACS control.

- *Helicopters were not required to file detailed flight plans and follow them.* Effective procedures were not established for communicating last minute changes or updates to the Army flight plans that had been filed.

- *F-15 pilots were not told to use non-HQ mode for helicopters.*

- *No procedures were specified to pass SITREP information to CFAC.* Helicopter flight plans were not distributed to CFAC and the F-15 pilots, but they were given to the F-16 squadrons. Why was one squadron informed, while another one, located right across the street, was not? F-15s are designed primarily for air superiority—high altitude aerial combat missions. F-16s, on the other hand, are all-purpose fighters. Unlike F-15s, which rarely flew low-level missions, it was common for F-16s to fly low-level missions where they might encounter the low-flying Army helicopters. As a result, to avoid low-altitude mid-air collisions, staff officers in F-16 squadrons requested details concerning helicopter operations from the JOIC, went to pick it up from the mail pickup point on the post, and passed it on to the pilots during their daily briefings; F-15 planners did not [108].

- *Inadequate training on the ROE was provided for new rotators.* Piper claims that OPC personnel did not receive consistent, comprehensive training to ensure they had a thorough understanding of the rules of engagement and that many of the aircrews new to OPC questioned the need for the less aggressive rules of engagement in what had been designated a combat zone [90]. Judging from these complaints (details can be found in [90]) and incidents involving F-15 pilots, it appears that the pilots did not fully understand the ROE purpose or need.

- *Inadequate training was provided to the F-15 pilots on visual identification.*

- *Inadequate simulator and spin-up training was provided to the AWACS crews.* Asynchronous evolution occurred between the changes in the training materials and the actual situation in the No-Fly-Zone. In addition, there were no controls to ensure the required simulator sessions were provided and that all members of the crew participated.

- *Handoff procedures were never established for helicopters.* In fact, no explicit or written procedures, verbal guidance, or training of any kind were provided to the AWACS crews regarding the control of helicopters within the TAOR [108]. The AWACS crews testified during the investigation that they lost contact with helicopters all the time but there were no procedures to follow when that occurred.

- *Inadequate procedures were specified and enforced for how the shadow crew would instruct the new crews.*

- *The rules and procedures established for the operation did not provide adequate control over unsafe F-15 pilot behavior, adequate enforcement of discipline, or adequate handling of safety violations.* The CFAC Assistant Director of Operations told the GAO investigators that there was very little F-15 oversight in OPC at the time of the shootdown. There had been so many flight discipline incidents leading to close calls that a group safety meeting had been held a week before the shootdown to discuss it. The flight discipline and safety issues included midair close calls, unsafe incidents when refueling, and unsafe takeoffs. The fixes (including the meeting) obviously were not effective. But the fact that there were a lot of close calls indicates serious safety problems existed and were not handled adequately.

  The CFAC Assistant Director of Operations also told the GAO that contentious issues involving F-15 actions had become common topics of discussion at Detachment Commander meetings. No F-15 pilots were on the CTF staff to communicate with the F-15 group about these problems. The OPC Commander testified that there was no tolerance for mistakes or unprofessional flying at OPC and that he had regularly sent people home for violation of the rules—the majority of those he sent home were F-15 pilots, suggesting there were serious problems in discipline and attitude among this group [90].

- *The Army pilots were given the wrong information about IFF codes to use in the TAOR.* As described above, this mismatch resulted from aynchronous evolution and lack of linkup (consistency) between process controls, i.e., the two different ATOs.

Reasons for the Flawed Control

Ineffective Control Algorithms Almost all of the control flaws at this level relate to the existence and use of ineffective control algorithms. Equipment and procedures were not coordinated between the Air Force and the Army to make sure communication channels were effective and that asynchronous evolution had not occurred. The last CTF staff member who appears to have actively coordinated rotary-wing flying activities with the CFAC organization departed in January 1994. No representative of the MCC was specifically assigned to the CFAC for coordination purposes. Since December 1993, no MCC helicopter detachment representative had attended the CFAC weekly scheduling meetings. The Army Liaison Officer, attached to the MCC helicopter detachment at Zakhu and assigned to Incirlik AB, was new on station (he arrived April 1994) and was not fully aware of the relationship of the MCC to the OPC mission [4]

Performance was not monitored to ensure that safety-critical activities were carried out correctly, that local adaptations had not moved operations beyond safe limits, and that information was being effectively transmitted and procedures followed. Effective controls were not established to prevent unsafe adaptations.

The feedback that was provided about the problems at the lower levels were ignored. For example, the Piper account of the accident includes a reference to helicopter pilots' testimony that six months before the shootdown, in October 1993, they had complained that the fighter aircraft were using their radar to lock onto the Black Hawks an unacceptable number of times. The Army helicopter pilots had argued there was an urgent need for the Black Hawk pilots to be able to communicate with the fixed-wing aircraft, but nothing was changed until after the accident, when new radios were installed in the Black Hawks.

Inaccurate Mental Models The Commander of the Combined Task Force thought that the appropriate control and coordination was occurring. This incorrect mental model was supported by the feedback he received flying as a regular passenger on board the Army helicopter flights, where it was his perception that the AWACS was monitoring their flight effectively. The Army helicopter pilots were using the Delta Point system to report their location and flight plans, and there was no indication from the AWACS that the messages were being ignored. The CTF Commander testified that he believed the Delta Point system was standard on all AWACS missions. When asked at the court martial of the AWACS Senior Director whether the AWACS crew were tracking Army helicopters, the OPC Commander replied:

> "Well, my experience from flying dozens of times on Eagle Flight, which that—for some eleven hundred and nine days prior to this event, that was—that was normal procedures for them to flight follow. So, I don't know that they had something written about it, but I know that it seemed very obvious and clear to me as a passenger on Eagle Flight numerous times that that was occurring" [90].

The Commander was also an active F-16 pilot who attended the F-16 briefings. At these briefings he observed that Black Hawk times were part of the daily ATOs received by the F-16 pilots and assumed that all squadrons were receiving the same information. However, as noted, the head of the squadron with which the Commander flew had gone out of his way to procure the Black Hawk flight information while the F-15 squadron leader did not.

Many of those involved at this level were also under the impression that the ATOs provided to the F-15 pilots and to the Black Hawks pilots were consistent, that required information had been distributed to everyone, that official procedures were understood and being followed, etc.

Coordination Among Multiple Controllers There were clearly problems with overlapping and boundary areas of control between the Army and the Air Force. Coordination problems between the services are legendary and were not handled adequately here. For example, two different versions of the ATO were provided to the Air Force and the Army pilots. The Air Force F-15s and the Army helicopters had separate control structures, with a common control point fairly high above the physical process. The problems were complicated by the differing importance of flexibility in flight plans between the two services. One symptom of the problem was that there was no requirement for helicopters to file detailed flight plans and follow them and no procedures established to deal with last minute changes. These deficiencies were also related to the shared control of helicopters by MCC and CFAC and complicated by the physical separation of the two headquarters.

During the accident investigation, a question was raised about whether the Combined Task Force Chief of Staff was responsible for the breakdown in staff communication. After reviewing the evidence, the hearing officer recommended that no adverse action be taken against the Chief of Staff because he (1) had focused his attention according to the CTF Commander's direction, (2) had neither specific direction nor specific reason to inquire into the transmission of info between his Director of Operations for Plans and Policy and the CFAC, (3) had been the most recent arrival and the only senior Army member of a predominantly Air Force staff and therefore generally unfamiliar with air operations, and (4) had relied on experienced colonels under whom the deficiencies had occurred [113]. This conclusion was obviously influenced by the goal of trying to establish blame. Ignoring the blame aspects, the conclusion gives the impression that nobody was in charge and everyone thought someone else was.

According to the official accident report, the contents of the ACO largely reflected the guidance given in the operations plan dated September 7, 1991. But that was the plan provided before the mission had changed. The accident report concludes that key CTF personnel at the time of the accident were either unaware of the existence of this particular plan or considered it too outdated to be applicable. The accident report states that "Most key personnel within the CFAC and CTF staff did not consider coordination of MCC helicopter activities to be part of their respective CFAC/CTF responsibilities" [4].

Because of the breakdown of clear guidance from the Combined Task Force staff to its component organizations (CFAC and MCC), they did not have a clear understanding of their respective responsibilities. Consequently, MCC helicopter activities were not fully integrated with other OPC air operations in the TAOR.

7.3.7 Conclusions from the Friendly Fire Example

When looking only at the proximate events and the behavior of the immediate participants in the accidental shootdown, the reasons for this accident appear to be gross mistakes by the technical system operators (the pilots and AWACS crew). In fact, a special Air Force task force composed of more than 120 people in six commands concluded that two breakdowns in individual performance contributed to the shootdown: (1) the AWACS mission crew did not provide the F-15 pilots an accurate picture of the situation and (2) the F-15 pilots misidentified the target. From the 21-volume accident report produced by the Accident Investigation Board, Secretary of Defense William Perry summarized the “errors, omissions, and failures” in the “chain of events” leading to the loss as:

- The F-15 pilots misidentified the helicopters as Iraqi Hinds.
- The AWACS crew failed to intervene.
- The helicopters and their operations were not integrated into the Task Force running the No-Fly-Zone operations.
- The Identity Friend or Foe (IFF) systems failed.

According to Snook, these four “causes” have been generally accepted by the military community as the explanation for the shootdown.

While there certainly were mistakes made at the pilot and AWACS levels as identified by the special Air Force Task Force and the four factors identified by the accident report and Perry were involved in the accident, the use of the STAMP analysis paints a much more complete explanation including: inconsistent, missing, or inaccurate information; incompatible technology; inadequate coordination; overlapping areas of control and confusion about who was responsible for what; a migration toward more efficient operational procedures over time without any controls and checks on the potential adaptations; inadequate training; and in general a control structure that did not enforce the safety constraints. While many of these factors result from “breakdowns in individual performance,” understanding the accident requires understanding the effects of the interactions among individual actions and decisions.

As noted earlier, STAMP treats an accident as a process. In this case, Army and Air Force operations adapted and diverged without communication and coordination. OPC had operated incident-free for over three years at the time of the shootdown. During that time, local adaptations to compensate for inadequate control from above had managed to mask the ongoing problems until a situation occurred where local adaptations did not work. A lack of awareness at the highest levels of command of the severity of the coordination, communication, and other problems is a key factor in this accident.

Concentrating on an event chain focuses attention on the proximate events associated with this accident and thus on the principle local actors, namely the pilots and the AWACS personnel. Treating the accident as a control problem using STAMP clearly identifies other organizational factors and actors and the role they played. Most important, without this broader view of the accident, only the symptoms of the organizational problems might have been identified and eliminated without significantly reducing risk of a future accident caused by the same systemic factors but involving different symptoms at the lower, technical and operational levels of the control structure.

More information on how to build multiple views of an accident using STAMP in order to aid understanding can be found in Chapter 8.

7.4 Applying the New Model

The use of STAMP points to new hazard analysis and prevention techniques, new risk assessment techniques, and new approaches to system monitoring and to the definition of safety metrics, all of which are described in Part III of this book.

STAMP focuses particular attention on the role of constraints in safety management. Accidents are seen as resulting from inadequate control or enforcement of constraints on safety-related behavior at each level of the system development and system operations control structures. Accidents can be understood, therefore, in terms of why the controls that were in place did not prevent or detect maladaptive changes, that is, by identifying the safety constraints that were violated at each level of the control structure as well as why the controls designed to enforce them were inadequate or, if they were potentially adequate, why the system was unable to exert appropriate control over their enforcement. The process leading to an accident (loss event) can be described in terms of an adaptive feedback function that fails to maintain safety as performance changes over time to meet a complex set of goals and values. Adaptation is critical in understanding accidents, and the adaptive feedback mechanism inherent in the model allows a STAMP analysis to incorporate adaptation as a fundamental system property.

We have found in practice that using this model helps us to separate factual data from the interpretations of that data: While the factors involved in accidents may be clear, their importance and the explanations for why the factors were present are often subjective. Our models are also more complete than most accident reports and other models. Each of the explanations for the incorrect FMS input of R in the Cali American Airlines accident described in Chapter 2, for example, appears in the STAMP analysis of that accident at the appropriate levels of the control structure where they operated. The use of STAMP not only helps to identify the factors but also to understand the relationships among them.

While STAMP models will probably not be useful in law suits as they do not assign blame for the accident to a specific person or group, they do provide more help in understanding accidents by forcing examination of each part of the socio-technical system to see how it contributed to the loss—and there will usually be contributions at each level. Such understanding should help in learning how to engineer safer systems, including the technical, managerial, organizational, and regulatory aspects.

To accomplish this goal, a framework for classifying the factors that lead to accidents was derived from the basic underlying conceptual accident model. This classification can be used in identifying the factors involved in a particular accident and in understanding their role in the process leading to the loss. The accident investigation after the Black Hawk shootdown identified 130 different factors involved in the accident. In the end, only the AWACS Senior Director was court martialed, and he was acquitted. The more one knows about an accident process, the more difficult it is to find one person or part of the system responsible, but the easier it is to find effective ways to prevent similar occurrences in the future.

STAMP is useful not only in analyzing accidents that have occurred but in developing new and perhaps more effective system engineering methodologies to prevent accidents. Hazard analysis can be thought of as investigating an accident before it occurs. Traditional hazard analysis techniques, such as fault tree analysis and various types of failure analysis techniques, do not work well for software and system design errors. Nor do they usually include organizational and management flaws. The problem is that these hazard analysis techniques are limited by a focus on failure events and the role of component failures in accidents and do not account for the complex roles that software and humans are assuming in high-tech systems and the indirect relationships between events and actions required to understand why the accidents occurred.

STAMP provides a direction to take in creating new hazard analysis and prevention techniques that go beyond component failure and are more effective against system accidents, accidents related to the use of software, accidents involving cognitively complex human activities, and accidents related to societal and organizational factors. Because in a system accident model everything starts from constraints, the new approach focuses on identifying the constraints required to maintain safety and then designing the system and operating conditions to ensure that the constraints are enforced. Such hazard analysis techniques augment the failure-based methods and encourage a wider variety of risk reduction measures than simply adding redundancy to deal with component failures. Chapter 9 provides detailed instructions on how to use STAMP to prevent accidents.

STAMP also can be used to improve performance analysis. Performance monitoring of complex systems has created some dilemmas. Computers allow the collection of massive amounts of data, but analyzing that data to determine whether the system is moving toward the boundaries of safe behavior is difficult. The use of an accident model based on system theory and the basic concept of safety constraints may provide directions for identifying appropriate safety metrics; determining whether control over the safety constraints is adequate; evaluating the assumptions about the technical failures and potential design errors, organizational structure, and human behavior underlying the hazard analysis; detecting errors in the operational and environmental assumptions underlying the design and the organizational culture; and identifying any maladaptive changes over time that could increase risk of accidents to unacceptable levels. Chapter 10 discusses approaches to accomplishing these goals.

Finally, STAMP points the way to very different approaches to risk assessment. Currently, risk assessment is firmly rooted in the probabilistic analysis of failure events. Attempts to extend current PRA techniques to software and other new technology, to management, and to cognitively complex human control activities have been disappointing. This way forward may lead to a dead end. Significant progress in risk assessment for complex systems will require innovative approaches starting from a completely different theoretical foundation—see Chapter 10.

Part III

Using STAMP in Systems Engineering and System Safety

Chapter 8

Analyzing Accidents and Incidents

[*This chapter is still in development and will be modified as we learn more about accident analyses using STAMP and further develop the approach, particularly the notation and models used.*]

A systems-theoretic view of accidents sees them as resulting from inadequate control or enforcement of constraints on safety-related behavior at each level of the system development and system operation control structures. Accidents can be understood, therefore, in terms of why the controls that were in place did not prevent or detect maladaptive changes, that is, by identifying the safety constraints that were violated at each level of the control structure as well as why the controls designed to enforce them were inadequate or, if they were potentially adequate, why the system was unable to exert appropriate control over their enforcement. The process leading to an accident (loss event) is described in terms of an adaptive feedback function that fails to maintain safety as performance changes over time to meet a complex set of goals and values. Adaptation is critical in understanding accidents, and the adaptive feedback mechanism inherent in the model allows a STAMP analysis to incorporate adaptation as a fundamental system property.

The use of a systems-theoretic accident model like STAMP does not lead to identifying single causal factors or variables. Instead it provides the ability to examine the entire socio-technical system design to understand the role each component plays in ensuring the safety constraints. This information can be used in an incident or accident investigation to identify the flaws in an existing structure and identify changes that will not simply eliminate symptoms but root causes. As will be seen in later chapters, it can also be used in a proactive way during system design, development, and operation to identify the flaws in the physical system design and safety controls that could lead to an accident and to use this information to design the physical system and controls to reduce the likelihood of an accident (Chapter 9), to establish metrics for detecting when the safety control structure is degrading and risk is becoming unacceptably high Chapter 10), to create new approaches to risk assessment (Chapter 10), and to assist in decision-making about changes in the system to determine whether they will increase risk (Chapter 9).

Hazard analysis has been described as analyzing accidents before they occur, so much can be learned about using STAMP in proactive prevention activities by understanding how it can be used to analyze accidents that have already happened. Using it for near miss or incident analysis can be a very effective way to perform root cause analysis and identify weaknesses in the safety control structure. Note that an accident investigation process is not being specified (a much larger topic), but only a way to document and analyze the results of such a process.

8.1 Applying STAMP to Accident Analysis

Accident analysis based on STAMP starts by determining the existing socio-technical safety control structure for the system involved. The accident process is described at each appropriate level of this control structure in terms of the safety constraints that were violated and why. Thus there will be multiple views of the accident, depending on the perspective and level from which it is being viewed.

The first step is to identify the hazard involved in the loss. Next, the hierarchical safety control structure related to the hazard is constructed and the constraints necessary to control the hazard are identified for each level. Then, starting from the technical process and using the proximate events and general application knowledge, any failures and dysfunctional interactions (including communication problems) involved in the loss are identified. For each constraint, a determination is made about why it was violated: either the constraint was never identified and enforced or the enforcement was inadequate. The classification in Figure 7.2 can be used in evaluating inadequate enforcement.

Any human decisions need to be understood in terms of (at least): the information available to the decision maker as well as any required information that was *not* available, the behavior-shaping mechanisms (the context and pressures on the decision making process), the value structures underlying the decision, and any flaws in the mental models of those making the decisions.

In general, the description of the role of each component in the control structure will include the following:

1. Safety Requirements and Constraints
2. Controls
3. Context:
   (a) Roles and Responsibilities
   (b) Environmental and Behavior Shaping Factors
4. Flaws in the Controlled Process
5. Dysfunctional Interactions, Failures, and Flawed Decisions, and Erroneous Control Actions
6. Reasons for Flawed Control Actions and Dysfunctional Interactions
   (a) Control Algorithm Flaws
   (b) Incorrect Process, Interface, or Mental Models
   (c) Inadequate Coordination among Multiple Controllers
   (d) Reference Channel Flaws
   (e) Feedback Flaws

The Friendly Fire shootdown described in Part II provides an example of a complex accident analysis using STAMP. While extremely complex, the accident involved only operations and thus the STAMP model includes only the control structure associated with operations. This chapter provides some examples of analyses based on STAMP that also involve technical system design and development flaws.

One difficulty in trying to analyze accidents from the information provided in accident reports is that much of the information needed to complete a STAMP model usually is not included because most are written from the perspective of an event-based model: The events are almost always clearly described and usually one or several of these events is chosen as the "root cause(s)," but the analysis of why those events occurred is usually incomplete. In addition, the analysis frequently stops after finding someone to blame, usually a human operator. The opportunity to learn important lessons for improving this and other systems is lost. In this chapter, STAMP is used to identify the additional questions that need to be answered to fully understand why the accident occurred and to maximize learning from the loss.

The first accident analyzed is the loss of the Ariane 5 on its first flight. This accident was chosen because the analysis at the technical level illustrates how the strict use of an event-chain model and direct links between events (as the Ariane report report did) can lead to ignoring important lessons that could be learned. Unfortunately, very little information is included in the report about how the system development and management of the development process led to the technical design errors.

The second example, the loss of a Milstar satellite being launched by a Titan IV/Centaur, provides an example of a STAMP analysis of the role played by system development and operations.

A final example of a water contamination accident in Canada provides a very complete example of the role of not only management but government and regulatory authorities in an accident and the corresponding analysis based on STAMP. This latter example, again because the information was included in the very comprehensive accident report, also demonstrates how the migration of an organization over time (adaptation) leads to accidents and the types of models that can be used to describe this migration.

8.2 The Ariane 5 Loss

On June 4, 1996, the maiden flight of the Ariane 5 launcher ended in failure: Only about 40 seconds after initiation of the flight sequence, at an altitude of only 2700 m, the launcher veered off its flight path, broke up, and exploded. The accident report describes the loss in terms of a chain of technical events, their inter-relations, and causes [71]. This chapter attempts to explain it using the STAMP model of accidents. Because the report focused on events as the explanation of the cause of the loss, it does not include all the information needed for a control focus on causality so the model is very incomplete. It is still informative, however, to put the available information into the STAMP format and to examine what additional information would have been useful to learn more from the accident. In addition, the analysis at the physical and software control levels (most of which can be gleaned from the report) provide an interesting illustration of the difference between an event-chain analysis and a systems-theoretic one.

To understand the accident, some background is necessary. The Flight Control System of the Ariane 5 is of a standard design, and much of the software built for the Ariane 4 was reused on the Ariane 5. The attitude of the launcher and its movements in space are measured by an Inertial Reference System (SRI, using the French acronym). The SRI has its own internal computer, in which angles and velocities are calculated on the basis of information from a strap-down inertial platform, with laser gyros and accelerometers. The data from the SRI are transmitted through the databus to the On-Board Computer (OBC), which executes the flight program and controls the nozzles of the solid boosters and the Vulcain cryogenic engine, via servovalves and hydraulic actuators.

In order to improve reliability, there is considerable redundancy at the hardware level: two SRIs operate in parallel, with identical hardware and software. One SRI is active and the other is in "hot" standby[1]. If the OBC detects that the active SRI has failed, it immediately switches to the standby unit, provided that this backup unit is functioning properly. Likewise, there are two OBCs, and a number of other units in the Flight Control System are also duplicated.

Exception handling was used to improve software reliability and fault tolerance. To oversimplify somewhat, when the computer hardware or operating system detects an error (called an *exception* or *fault*) in the application software, it usually stops the execution of the application software. There are good reasons to stop because most of the time continuing will result in bad consequences. When *exception handling* is used in the application software, instead of just stopping its execution, the hardware or operating system can signal to the application software that a fault or exception has occurred. This behavior is called *raising an exception*, and the type of exception raised provides some limited information about what type of error occurred. The types of exceptions that can be detected are primarily faults confined to a single instruction, such as an overflow (e.g., bits have been lost in a variable value as the result of an arithmetic operation or from assigning a variable value to a location that does not have enough bits to hold the entire value), an attempt to divide by zero, or an attempt to access a variable in protected memory. In order to *handle* the signaled exception, the application program must include special exception-handling code that has been designed to detect the signal and perform some repair. If no exception-handling code is included or if the repair attempt is unsuccessful, execution of the application software usually is stopped. Alternatively, exception-handling software may contain logic to continue execution, perhaps simply skipping the erroneous instructions.

The losses from the accident included the $5 billion payload, which in accordance with common practice for test flights was not insured, as well as other unknown financial losses. The accident also had repercussions in the form of adding two qualification flights that had not been planned, pushing back commercial flights by more than 2.5 years, and extending Ariane 4 operations until 2003 (it was originally scheduled to be phased out in 2000).

8.2.1 The Launch Vehicle

For the new model, the loss event (accident) is first described with respect to the components of the technical process, in this case the Ariane launch vehicle. The loss involved the following dysfunctional interactions in the launch vehicle:

- **Ariane 5:** A rapid change in attitude and high aerodynamic loads stemming from a high angle of attack create aerodynamic forces that cause the launcher to disintegrate at 39 seconds after the command for main engine ignition (H0).

- **Nozzles:** Full nozzle deflections of the solid boosters and Vulcain main engine lead to an angle of attack of more than 20 degrees.

[1] A hot standby operates in parallel with the primary unit but either it does not provide outputs to the databus or its outputs are ignored.

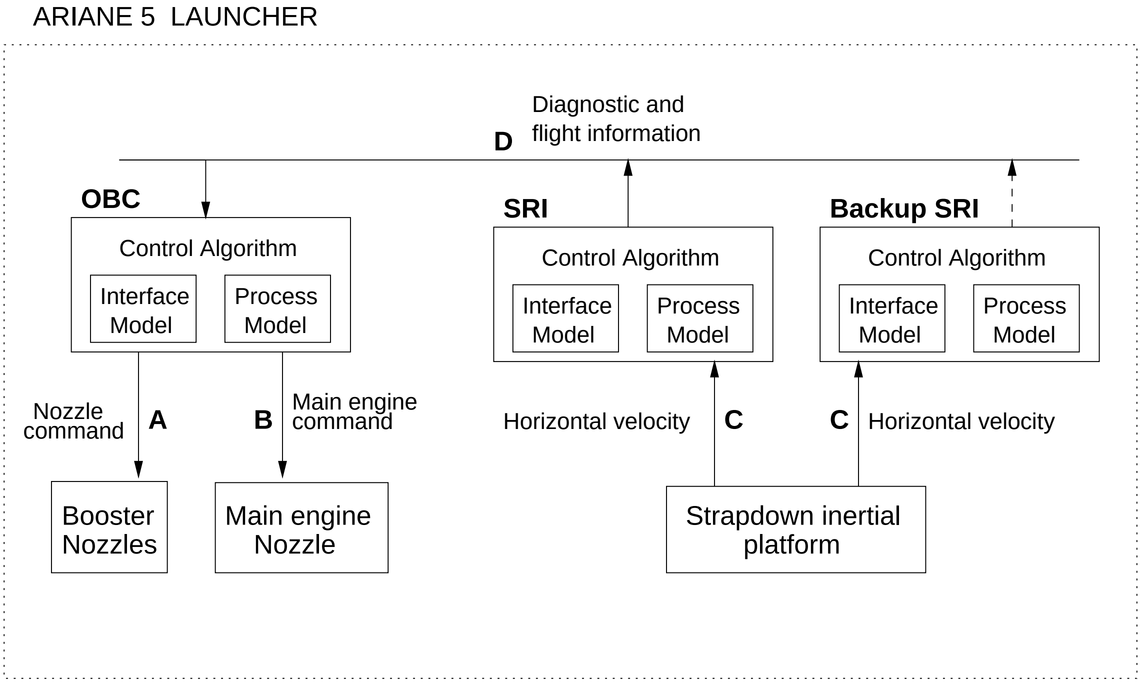


Figure 8.1: Lowest Level Control Loop

- **Self-Destruct Systems:** The self-destruct system is triggered (as designed) by the separation of the boosters from the main stage at an altitude of 4 km. and a distance of 1 km. from the launch pad.

The system hazard violated and safety constraint not enforced on the process in this case were:

Hazard: Change of attitude that causes disintegration of launcher.

System Safety Constraint: The launcher must not go outside a safe envelope[2].

The next step is to take each part of the technical control loop and evaluate why the system safety constraint on the process state was not enforced. The factors identified in Figure 7.2 can be used to assist in this process. The analyst then moves upward in the operations and development control structures to determine at each level the safety constraints that were violated and why.

8.2.2 The Technical Control Loop

For the Ariane 5, the lowest-level (technical) control loop is shown in Figure 8.1. The OBC (On-Board Computer) and SRI (Inertial Reference System) along with its backup are the relevant system components for which safety constraints and their violation need to be examined:

[2] This constraint would most likely be specified using a more detailed description of the safe envelope, but the details are unknown to me.

Inertial Reference System (SRI)

The SRI measures the attitude of the launcher and its movements in space on the basis of information provided by a strap-down inertial platform. The requirements stated that in the event of any type of exception, "the failure should be indicated on the databus, the failure context should be stored in an EEPROM memory . . . and finally the SRI processor should be shut down."

The goal of the backup SRI was to take over in case the primary SRI was unable to send guidance information. The software in both the primary and backup SRIs was identical and therefore provides protection only against hardware failures.

SRI Safety Constraints Violated: (1) The SRI must continue to send guidance information as long as it can get the necessary information from the strapdown inertial platform; (2) the SRI must not send diagnostic information in a form that could be misinterpreted as flight data.

Unsafe Behavior: At 36.75 seconds after H0, the SRI detected an internal error in a non-critical function and turned itself off (as it was designed to do) after putting diagnostic information on the databus.

Control Algorithm Flaw: The algorithm calculates the horizontal bias (an internal alignment variable used as an indicator of alignment precision over time) using the horizontal velocity input from the strapdown inertial platform (C in Figure 8.1). Conversion of this variable from a 64-bit floating point value to a 16-bit signed integer led to an unhandled overflow exception while calculating the horizontal bias. No instructions for handling this exception were included in the software, and the computer shut itself down after putting diagnostic information on the databus.

Process Model: The SRI process model did not match the Ariane 5. Instead, it was based on Ariane 4 trajectory data. The software was reused from the Ariane 4, where the velocity variables are lower than the Ariane 5. The horizontal bias variable on the Ariane 4 does not get large enough to cause an overflow.

Backup Inertial Reference System (SRI)

SRI Safety Constraint Violated: The backup SRI must continue to send attitude and control data as long as it can get the necessary information from the strapdown inertial platform.

Unsafe Behavior and Dysfunctional Interactions: At 36.7 seconds after H0, the backup SRI detected an internal error in a non-critical function and turned itself off (as it was designed to do).

Process Model: The SRI process model did not match the Ariane 5. As in the primary SRI, the model is based on Ariane 4 trajectory data.

Control Algorithm: This control algorithm was the same as used in the SRI, i.e., it calculated the horizontal bias using the same horizontal velocity input from the strapdown inertial platform. Conversion from a 64-bit floating point value to a 16-bit signed integer led to the same unhandled overflow exception while calculating the horizontal bias. Because the algorithm was the same in both SRI computers, the overflow results in the same behavior, i.e., shutting itself off.

Process Model Flaws: The SRI process model did not match the Ariane 5. As in the primary SRI, the model is based on Ariane 4 trajectory data, which again assumed smaller horizontal velocity values than possible on the Ariane 5.

On-Board Computer (OBC)

The on-board computer executed the flight program and controlled the nozzles of the solid boosters and the Vulcain cryogenic engine, via servovalves and hydraulic actuators.

OBC Safety Constraints Violated: (1) Commands from the OBC to the nozzles must not result in the launcher operating outside its safe envelope; (2) The OBC must know the status of the SRI and backup SRI at all times and which computer input to use; (3) the OBC must not confuse guidance data with other inputs received.

Unsafe Control Commands (A and B): A control command was sent to the booster nozzles and later to the main engine nozzle to make a large correction for an attitude deviation that had not occurred.

Control Algorithm Flaws: This algorithm read SRI diagnostic information (indicating it had an internal error and was shutting down) from the databus, interpreted it as guidance data, and used it for flight control calculations. With both the SRI and the backup SRI shut down and therefore no possibility of getting correct guidance and attitude information, the Ariane 5 loss was inevitable. But if the backup SRI *had* continued to operate, reading and using diagnostic information from the SRI in flight-control calculations might have led to the loss on its own so this error is an important one. There is no information in the accident report to indicate whether the OBC software included a check for the diagnostic information and it was simply unsuccessful at recognizing it, whether such a check was omitted, or whether the flaw was in the design of the diagnostic message itself. A basic safety design principle is that diagnostic and other information should be sent in a form that is easily distinguishable from (cannot be confused with) normal data [66]. In addition, no information is provided about whether the OBC algorithm included checks for out-of-range data, another standard defensive programming strategy.

Flight Information and Diagnostic Inputs (D): The diagnostic information was placed on the bus by the SRI. The report does not indicate whether (2) this information was intended for the backup SRI to indicate that the primary SRI computer had ceased operation and that the backup SRI should take over (i.e., to move from hot standby to primary and to start sending guidance information to the OBC) or, (2) if the backup SRI was already sending data, to indicate to the OBC to start using the backup SRI data.

Process (Spacecraft) Model Flaws: The model of the current launch attitude was incorrect, i.e., it did not match the spacecraft attitude because it assumed an attitude deviation had occurred that had not. The incorrect process model led to incorrect control commands being sent to the nozzles. The internal process model of the inferred spacecraft state became inconsistent with the actual spacecraft state through an incorrect update of the internal model (the spacecraft attitude) using the contents of an SRI diagnostic message that was incorrectly interpreted as flight information.

Interface Model Flaws: This model was incomplete or incorrect (not enough information is provided in the report to determine which), i.e., it did not include the SRI diagnostic message formats available on the databus or it did not contain adequate information about these messages. It is unclear from the accident report whether the OBC was ever intended to read this type of message, although it probably was. The OBC Interface Model also included a mismatch between its model of the SRI state and the true SRI state—it thought the SRI was active when it actually was shutdown or it would not have used its inputs.

Unfortunately, almost no information is included in the accident report about the safety-related flaws in the OBC software because an argument is made that with the loss of the navigation and guidance information from the SRIs, the loss was inevitable. While true, this narrow view rules out the possibility of learning about important software flaws: Even if the backup SRI had continued to work, interpreting the diagnostic message as flight information might have led to the loss of the launcher anyway. Several accidents or near misses have resulted from similar misinterpretation of informational messages by software, for example, the interpretation of a sanity check message as an indication of a Russian missile attack by a NORAD computer [66].

The requirement for the SRI software to send the diagnostic information was in the SRI software requirements and was correctly satisfied. What is not included in the report is (1) whether the OBC software requirements contained a requirement to process the diagnostic data and, if so, whether the programmers were told to treat the message as an indication that the SRI had failed and to start using the Backup SRI or to ignore the message (i.e., the message was intended only for the Backup SRI), (2) whether there was an interface specification (or at least whether one was available when the OBC software requirements and design were being developed), (3) if there was an interface specification, whether it was used but it did not contain a description of the diagnostic data being put onto the databus, or (4) whether the interface specification was correct but the OBC software developers did not use it or interpreted it incorrectly. These questions need to be investigated in order to determine the flaws in the development process that led to these software flaws and to prevent them in the future.

There is also no information about whether potentially unsafe behavior (such as the limits of safe commands to the servovalves and hydraulic actuators) was identified to the OBC software developers and whether they attempted to check commands before sending them to determine whether they lay outside a safe envelope. Alternatively, such checks may have been included but were based on Ariane 4 trajectory data, as was the SRI software. This question would have been an important one to answer for full understanding of the accident process.

8.2.3 System Operations Control Structure

The report does not mention anything about operators and presumably they played no role in the accident.

8.2.4 System Development Control Structure

The next step is to examine the higher levels of control to understand why these control flaws occurred: why they were introduced, why they were not found before use, and why they were not successfully handled during operations. For each flaw and for each level of the operations and development control structures, we need to understand the decisions involved and to evaluate them in terms of the information available and the basis for the decision. Why were the decisions made? Why did they turn out to be flawed, i.e., what was the basic flaw in the decision-making process and mental models of the decision makers that led to the result? Why, at each level of the control structure, were the controls inadequate to prevent the accident?

The accident report is almost totally silent about the development control structure and overall management structure and how organizational and management factors might have impacted the process leading to the accident. For example, it does not describe the allocation of responsibility and authority for safety nor the project management and system safety controls in effect during development. What safety policy and standards were applied? What standards were applied to decisions about reusing software from the Ariane 4? Was a hazard analysis performed? If so, how were the results communicated and used?

Enough information is included, however, to give some idea of what an analysis of the accident might contain using the new model. Where the information necessary to evaluate the accident from the STAMP perspective was not available, the additional necessary questions that might have been investigated can be identified.

The following includes the information that was either in the report or could be assumed from what was included. A generic control structure is shown in Figure 8.2.

8.2.5 Software Engineering

Two sets of software engineers (and system engineers) are involved: those who wrote the original Ariane 4 software and those working on the Ariane 5 control systems. The report does not indicate whether there were overlaps in personnel for the two development processes (most likely there was).

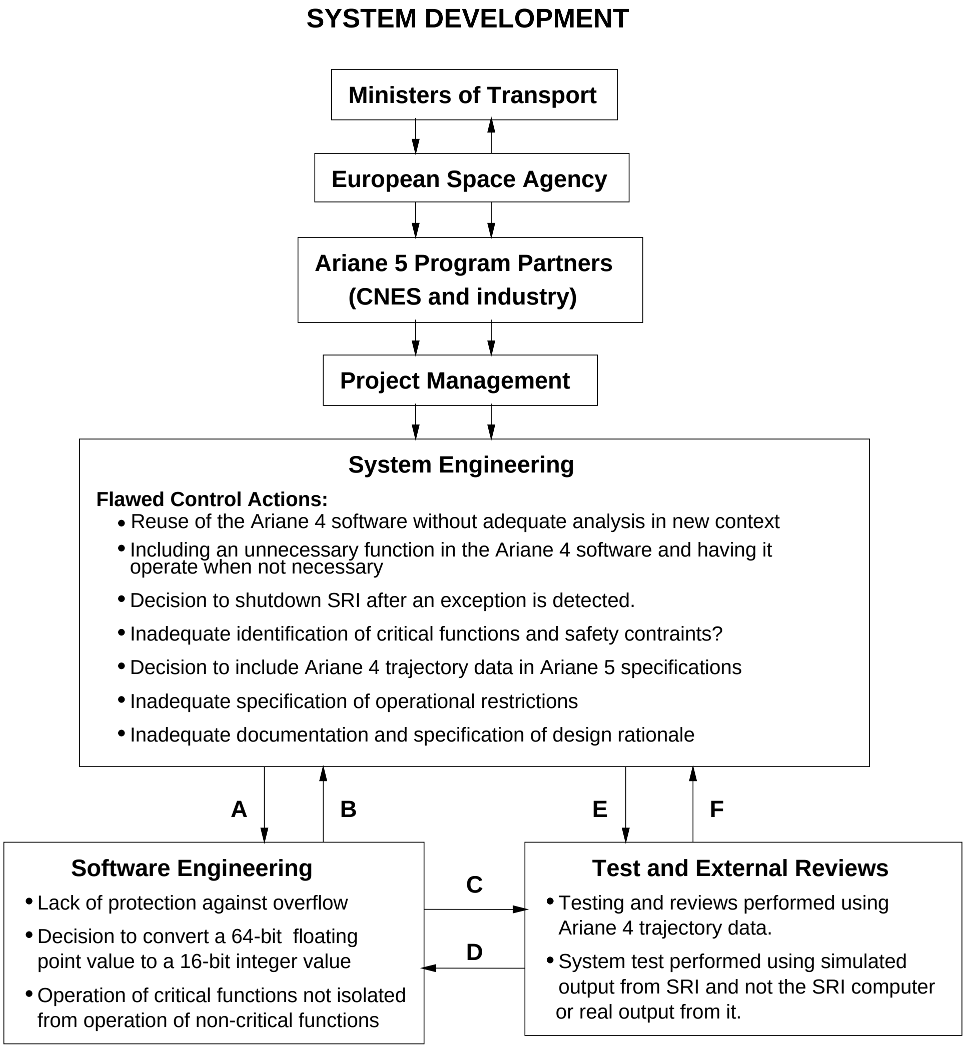
There were three important flaws in the SRI software related to the loss: (1) the lack of protection against overflow for some variables, (2) the conversion of a 64-bit floating point value to a 16-bit integer value, and (3) the lack of an effective firewall between the critical and non-critical software-implemented functions so that a flaw in non-critical software could not affect the operation of the critical functions.

1. Lack of Protection Against Overflow

The horizontal velocity of the Ariane 4 cannot reach a value beyond the limit of the software (the value will always fit in a 16-bit integer variable), but the acceleration and trajectory of the Ariane 5 during the first 40 seconds leads to a buildup of horizontal velocity five times more rapid than that of the Ariane 4.

The report states that during software development, an analysis was performed on every operation that could give rise to an exception, including an operand error, to determine the vulnerability of unprotected code. In particular, the conversion of floating point values to integers was analyzed and operations involving seven variables were determined to be at risk of leading to an operand (overflow) error. Exception handlers were written for four of the seven variables but not for the other three, including the horizontal bias variable.

The report says that this decision to leave three variables unprotected was made jointly by the Ariane project partners at several contractual levels. Presumably this decision was made during the original software development for the Ariane 4, but the report does not say whether the decision was revisited during design of the Ariane 5. Apparently, it was not.


SYSTEM DEVELOPMENT
Ministers of Transport
European Space Agency
Ariane 5 Program Partners (CNES and industry)
Project Management
System Engineering
Flawed Control Actions:
• Reuse of the Ariane 4 software without adequate analysis in new context
• Including an unnecessary function in the Ariane 4 software and having it operate when not necessary
• Decision to shutdown SRI after an exception is detected.
• Inadequate identification of critical functions and safety contraints?
• Decision to include Ariane 4 trajectory data in Ariane 5 specifications
• Inadequate specification of operational restrictions
• Inadequate documentation and specification of design rationale
A
B
E
F
Software Engineering
• Lack of protection against overflow
• Decision to convert a 64-bit floating point value to a 16-bit integer value
• Operation of critical functions not isolated from operation of non-critical functions
C
D
Test and External Reviews
• Testing and reviews performed using Ariane 4 trajectory data.
• System test performed using simulated output from SRI and not the SRI computer or real output from it.

Communication Flaws (reference channel and measuring channel):

A: No requirement to separate critical from non-critical functions.
Were hazards and critical functions identified to the developers???
Ariane 4 trajectory data provided to developers (not Ariane 5)
Operational restrictions not passed to developers

B: (inadequate information)

C: NecessInformation about decision to leave three variables unprotected not provided to reviewers

D: Test results incomplete, review reports incomplete

E: Operational restrictions not passed to testers or reviewers
Ariane 4 trajectory data provided to testers and reviewers (not Ariane 5)

F: Test reports incomplete, review results incomplete

Figure 8.2: System Development Control Structure

Why was this decision made originally?

1. The logic necessary to satisfy all requirements completely would take more than the maximum workload target of 80% set for the SRI computer. An 80% load factor is usually considered the maximum acceptable for embedded systems. The extra margin is usually kept for several reasons, including to allow for mistakes in computation of performance requirements (i.e., to allow a margin of error), for adding new requirements and functions, and for maintenance and upgrade. To remove this condition requires either reducing the requirements (not considered in the accident report but it should have been) or changing computers (also not mentioned). The latter is usually impractical without seriously impacting schedule and budget. Often because of the length of the development process for spacecraft, hardware that seemed perfectly adequate in the early design stages or was the most appropriate at hardware selection time is obsolete by the time the system is ready for first launch but at that time it is too late to make changes. The problem is compounded when the software and presumably the hardware is reused for new upgrades or designs like the Ariane 5.

2. The report says that analysis indicated that the three unprotected variables were either physically limited or there was a large margin of safety. It also states "there is no evidence that any trajectory data was used to analyze the behavior of the three unprotected variables," It is not clear whether this latter statement is with respect to the original development team or the Ariane 5 developers (which may have included overlaps). The original designers might have had no idea that the software would be reused in a different environment where the environment would falsify the assumptions about the margins of safety. Assuming the original developers could have no idea about any future reuse decisions, the problem lies in reusing the software without redoing the analysis. Were the analysis and the decision making involved in it recorded so that changes in the environment could trigger a re-evaluation? What types of analysis were performed when a decision to reuse the SRI software on the Ariane 5 was made? Software reuse has been implicated in many accidents—any reuse must involve careful evaluation of the differences in the new environment that may violate assumptions made during the software development. The ability to perform this new analysis requires that the original design decisions and design rationale have been recorded or can be regenerated easily.

Another consideration not included in the accident report is that even if reuse had not been considered during the original software development, defensive programming practices would suggest that protection be provided in the software against unexpected events. While writing exception handlers might take space, the insertion of in-line checks for potential overflow when moving values into smaller memory locations takes minimal space and time and is fairly standard practice for real-time software. In fact, the SRI software was written in Ada, which provides built-in facilities that can be used to perform this check easily. There is no mention in the report about whether such a check was attempted in the code and failed to detect the impending overflow or was not included. Checking for potential overflow is so simple and fast in Ada and involves so little additional space requirements, there does not seem to be any obvious reason why it would not be performed even if it had been determined through analysis (which can always potentially be incorrect) that the variables would never be that large. The same arguments apply to checks by the OBC software for out-of-range attitude values provided by the SRI; the report is again silent about whether these might have been included.

The use of such defensive programming strategies is even more important if there is any chance that the process being controlled may change or the software may be reused in a different system. A more complete accident investigation (based on a system-theoretic accident model like STAMP) might have examined the defensive programming practices used in software development, which then would have led to an examination of the programming standards that were used on the project.

2. Decision to Convert a 64-bit Floating Point Value to a 16-bit Integer Value.

The software contains code to convert the horizontal bias value originally stored as a 64-bit floating point value to a 16-bit integer value. A significant omission in the report is why the conversion from 64-bit floating point to 16-bit integer was done at all. The potential loss of information in such a conversion would seem on the surface to be a mistake (although it is not called out as such in the accident report) and would again violate standard defensive programming practice unless a very good reason were involved. One reason for the conversion might be the need for *mixed-type* arithmetic, although conversion of all values to floating point would be standard safe practice in that case. Another reason might be related to the 80% load factor criterion applied to memory usage. Without understanding why the conversion was included, fully understanding why the accident occurred is not possible. It is odd that the accident report does not question this programming decision. A similar software flaw almost resulted in the loss of a Telstar satellite while being launched from the Space Shuttle. As noted above, once such a decision is made, there is little reason not to include checks to detect potential overflow before it occurs.

3. Lack of an Effective Firewall between the Critical and Non-Critical Software-Implemented Functions.

A basic design principle in any system with critical functions is to design such that non-critical functions cannot interfere with the provision of the critical functions. Shutting down the computer in the event of an exception in a non-critical function seems to violate safe practices even if there was assumed to be another backup SRI to provide the necessary functionality. There is not enough information in the accident report to understand the reason for this flawed software design. Potential explanations include software engineers not being informed about which functions were critical, not understanding the need for firewalls, inadequate firewall design, etc. Again, it is surprising that this decision was not discussed in the report, perhaps as part of a larger investigation into the programming standards used on the project.

8.2.6 Internal and External Reviews

Once again, two sets of reviewers are involved: those for the Ariane 4 development and those for the Ariane 5. The report says that the Ariane 5 reviews "included all the major partners in the Ariane 5 program" so Ariane 5 software reviews *did* take place. For some reason, these reviews did not uncover the problems. Unfortunately, the report provides no information about the review process and very little information about why this process failed to uncover the multiple design flaws.

The only clue as to the deficiencies in the review process come from the recommendations that suggest including external (to the project) participants when reviewing specifications, code, and justification documents and to make sure that these reviews consider the substance of arguments, rather than check only that verifications have been made. The latter seems to reflect the common problem of reviewers and assurance personnel simply ticking off on checklists that activities have been performed without reviewing their quality and substance.

It is clear that the reviewers did not have the necessary information to review the decision to leave three variables unprotected. The report states that the justification of the decision was not included directly in the source code, but it is unclear whether it was documented elsewhere. However, the report also states that "Given the large amount of documentation associated with any industrial application, the assumption, although agreed, was essentially obscured, though not deliberately, from any external review." The next chapter provides an approach to specification that includes design rationale and enhances the ability to find this information when it is needed.

More important, the report states that neither the Ariane 5 SRI requirements nor specification included Ariane 5 trajectory data. Instead, Ariane 4 trajectory data was included. The report states that this was a choice—it was jointly agreed upon by the developers—and was not simply an inadvertent omission. There is no explanation given, although the decision is so strange that there must have been some rationale for it. Whenever software is reused, the assumptions about the environment of the current system must reflect the new environment. A detailed analysis of differences in the two environments and their impact on the safety of the reused software should be required.

The quality of the review process is, of course, greatly influenced by the quality of the specifications. Incorrect assumptions and missing information for reviews lead to process and mental models that contain a mismatch between the development and the use environment, i.e., the model of the controlled process and its environment used by the developers, reviewers, and testers is inconsistent with the actual process and environment.

No information is provided about the review process for the other flaws in the technical control loop leading to the loss. It would seem, for example, that potentially dangerous coding practices (such as conversion from a floating point value to an integer) should always trigger a careful review. Were such reviews performed? Who was included? What information did the reviewers have? What information did they *not* have that might have helped them to find the system and software design flaws?

8.2.7 Testing

A conclusion that poor testing was the "cause" of an accident is always suspect. After the fact, it is always easy to find a test case that would have uncovered a known error, but it is usually difficult to prove that the particular test case would have been selected beforehand even if testing procedures were changed. By definition, the cause of an accident can always be stated as a failure to test for the condition that was, after the fact, found to have led to the loss. However, in this case, there do seem to be omissions that reflect some poor decisions related to testing, particularly with respect to the accuracy of the simulated operational environment.

Safety Constraints Violated: The test environment must match the operational environment as closely as possible.

Dysfunctional Interactions, Inadequate Control Actions, and Flawed Decisions: Two problems are noted in the accident report.

1. Simulations and Test of the SRI Software were Performed using Ariane 4 Trajectory Data.

No test or simulation was performed to verify that the SRI software would behave correctly when being subjected to the countdown and flight time sequence and the trajectory of the Ariane 5. This testing inadequacy is not surprising given that the specifications included only Ariane 4 trajectory data. Again, there was a process model mismatch between the model of the operational environment used in test and the real process.

2. System Test and System Simulations were Performed using Simulated Output from the SRI and not the SRI itself or its Detailed Simulation

The two SRI computers were not present during system test. The precision of the navigation software in the flight control computer (OBC) depends on the precision of the SRI measurements, but in the system test facility this precision could not be achieved by the electronics creating the test signals. The precision of the simulation may be further reduced because the base period of the SRI is 1 millisecond versus 6 milliseconds in the simulation at the system test facility. This difference adds to the complexity of the interfacing electronics. However, while physical laws preclude the testing of the SRI as a blackbox in the flight environment, the report states that ground testing is possible by injecting simulated accelerometric signals in accordance with predicted flight parameters, while also using a turntable to simulate launcher angular movements. This test was not done.

The report identifies the reason for the lack of such testing as a set of faulty assumptions:

- Assuming that the SRI was fully qualified by unit testing and by previous use on Ariane 4. This conclusion reflects a confusion between reliability and safety and the need to thoroughly test and analyze reused software in the new system context.
- Assuming that the simulation of failure modes is not possible with real equipment, but only with a model. This argument assumes hardware-type failures, not logic errors—the latter should be detectable during system simulation testing. This flaw in the decision making reflects misunderstanding of the difference between hardware failures and software "failures".[3]
- Assuming that inadequate precision precluded effective ground testing.

A general principle in testing aerospace systems is to fly what you test and test what you fly. The test and simulation processes need to reflect the environment accurately. Although following this principle is often difficult or even impossible for spacecraft, there seems to be no excuse for not including Ariane 5 trajectory data in the specifications and simulations. The report says that

> It would have been technically feasible to include almost the entire inertial reference system in the overall system simulations which were performed. For a number of reasons, it was decided to use the simulated output of the inertial reference system, not the system itself or its detailed simulation. Had the system been included, the failure could have been detected [71].

[3] The widespread application of the term "failure" to software design errors encourages these types of misunderstandings.

8.2.8 System Engineering

Very little information is provided about the system engineering process, although this obviously was the source of almost all of the technical flaws leading to the accident. However, there is enough information to examine some of the critical decisions involved in the process and to identify potential improvements in the control structure in order to reduce flawed engineering decisions and procedures. The flaws in the SRI process models and control algorithm can be traced back to the system engineering decision to reuse Ariane 4 software without an adequate safety analysis of its potential impact on the Ariane 5, flawed (inadequate) fault-tolerant and fail-safe design, and providing inadequate specifications and information to the development and assurance processes.

Because the flaws are common in many systems today, they are discussed here in greater detail that is strictly necessary simply to illustrate the use of the new accident model.

1. Decision to Reuse Ariane 4 Software without Effective Safety Analysis:

Information Available: The report is silent about any analysis that might have been generated to provide information about the safety of this decision.

Context and Influences: Competitive and budgetary pressures are not mentioned in the public report, but they could explain the reuse and other decisions. The launch market is potentially very large (estimated to be $60 billion in the next decade) and competitive. Europe was struggling to defend its greater than 50% market share from competitors such as ILS (a joint venture of Lockheed Martin, Russian, and Ukrainian firms using the Proton rocket), Boeing with its Delta Rocket and Sea Launch project, and other possible challengers from Japan, China, and Brazil. Did this competition affect the development schedule and critical decisions such that Ariane 4 software was reused and proper analysis and testing was not performed?

Basis for the Decision: The rationale for the decision to reuse the SRI software from Ariane 4 is described only as "for commonality reasons" in the report, which could be interpreted too many ways to get much information out of the statement.

Reuse (and the related use of commercial off-the-shelf software—COTS) is common practice today in embedded software development and often imposed on the developers by managers who think that money will be saved. A very common misconception is that because software operated safely in other applications, it will be safe in the new one. This misconception arises from a confusion between software reliability and software safety. Although component reliability is an important factor in component failure accidents, it is irrelevant for system accidents where none of the components involved fail. Almost all software-related accidents have been system accidents and involved flaws in the software requirements (the specified blackbox behavior) and not implementation errors—that is, the software satisfied its requirements and operated "correctly."

Blackbox behavior of a component can only be determined to be safe by analyzing its effects on the system in which it will be operating, i.e., by consideration of the specific operational context. The fact that software was used safely in another environment provides no information about its safety in the current one. In fact, reused software is probably less safe as the original decisions about the required software behavior were made for a different context. This fact may account for the large number of incidents and accidents that have involved reused software.

The philosophy of the Ariane 5 project, as stated in the accident report, that it was not wise to change software that had worked well on the Ariane 4 unless it was proven necessary to do so, is well founded: Errors are often introduced when software is changed, particularly by those who did not originally write it. However, in this case, there is a tradeoff with safety that needs to be carefully evaluated. The best solution may not have been to leave the software as it was but to rewrite it—such a decision depends on the type of change required, the specific design of the software, and the potential effect of the feature on system safety. In this case, the cost of the decision to reuse the Ariane 4 software (in terms of both money and program delays) was much greater than the cost of doing a proper analysis. System engineering should include in the risk management and system engineering tradeoff process an evaluation of the potential risk of reusing components (or buying them off-the-shelf) versus creating new ones. If the cost of the analysis for reused (or COTS) code is prohibitively expensive or beyond the state of the art, then redeveloping the software or rewriting it may be the appropriate decision.

The implication of this discussion is not that software cannot be reused, but that a safety analysis of its operation in the current system context is mandatory. (Note that testing is not adequate to accomplish this goal). For this type of analysis to be both technically and financially practical, it is important that reused software contain only the features (specified software behaviors or requirements) necessary to perform the critical functions. Note, however, that COTS software often is built to include as many features as possible in order to make it commercially useful in a large variety of systems.

If a decision is made to reuse software, system engineering should analyze it for safety within the system being developed. This goal can be accomplished by tracing the identified system hazards to the software (or hardware) component being considered and determining whether the component is capable of producing this behavior. If it is, protection is required in the form of changes to the system or component design.

2. Decision to have the SRI continue to execute the alignment function after takeoff.

Information Available: Unknown

Context and Influences: Unknown

Basis for Decision: Unknown

Why Decision was Flawed (Unsafe): The alignment function computes meaningful results only before liftoff—during flight, the function serves no purpose:

> Alignment of mechanical and laser strap-down platforms involves complex mathematical filter functions to properly align the x-axis to the gravity axis and to find north direction from Earth rotation sensing. The assumption of preflight alignment is that the launcher is positioned at a known and fixed position. Therefore, the alignment function is totally disrupted when performed during flight, because the measured movements of the launcher are interpreted as sensor offsets and other coefficients characterising [*sic*] sensor behaviour [71]

Normal alignment takes 45 minutes or more. The alignment function in the Ariane 4 was designed to cope with the unlikely event of a hold in the Ariane 4 countdown, i.e., between liftoff minus 9 seconds when flight mode starts in the Ariane 4 SRI and liftoff minus 5 seconds when certain events are initiated in the launcher that would take hours to reset. The alignment function continues to operate for 50 seconds after the start of flight mode to allow enough time for the ground equipment to resume control in case of a hold. Without a hold, the additional operation of this function was useless. This design allowed the countdown to be restarted after a hold and a short launch window could still be used. The feature was used once in 1989 on Flight 33 of the Ariane 4. The Ariane 5 has a different preparation sequence, and thus cannot use the feature at all.

Again, while there may be good reason for not wanting to rewrite or change software that is being reused (and potentially introduce errors), all such functions at a minimum need to be analyzed for their impact on safety when reused. The report does question the rationale behind allowing the alignment function to operate after the launcher has lifted off by including a recommendation that the alignment function of the SRI be switched off immediately after liftoff. More generally, it recommends that no software function should execute during flight unless it is needed. It does not, however, question the decision to include the feature in the Ariane 4 software originally (see below) nor the decision not to eliminate the feature from the Ariane 5 software.

The flawed decision here is related to a general problem often associated with software-related accidents. Behavior that is safe while the controller or plant is operating in one mode may not be safe in a different mode. In this case, providing alignment functions was unsafe only after liftoff, and the control algorithm should have checked for this mode change. Startup modes and behavior tend to be particularly error-prone and unsafe. Other accidents have occurred where functions were incorrectly operating in aircraft when there was *weight-on-wheels* or, alternatively, incorrectly operating when there was *not* weight-on-wheels, i.e., the process model is incomplete and does not match the process. A likely causal factor in the Mars Polar Lander loss was in software that was operating when it was not needed. This common accident factor is discussed further with respect to hazard analysis in Chapter 9.

3. Decision to include an unnecessary but useful feature in the original Ariane 4 SRI software:

Information available: Unknown

Context and Influences: Including the ability to reset quickly in the event of a hold in the Ariane 4 countdown was undoubtedly a potentially useful feature, as would many other features that could be included in inertial reference system software. There is always a tendency when designing systems with software components for the introduction of *creeping featurism* or the addition of more and more useful features. Including features that are not critical, however, adds to software and system complexity, makes testing and analysis harder, and degrades the safety of reusing the software (or at least adds a very costly, difficult, and error-prone analysis requirement when it is reused).

Why Decision was Flawed: While the accident report does question the advisability of retaining the unused alignment function from the Ariane 4 software, it does not question whether the Ariane 4 software should have included such a non-required but convenient software function in the first place. Outside of the question of its effect on reuse (which may reasonably not have been contemplated during Ariane 4 software development), a tradeoff was made between perhaps delaying a launch and simplifying the software. The feature was used only once in Ariane 4 launches. The more features included in software and the greater the resulting complexity, the harder and more expensive it is to test, to provide assurance through reviews and analysis, to maintain, and to reuse in the future.

System engineering needs to include in the risk management and system engineering tradeoff processes a realistic assessment of the risk of including unnecessary but useful software functions.

4. Decision to shut down the SRI after an exception:

The SRI is designed to shut itself down after an error is encountered. Restart of the inertial reference system, once it has shut down, is not feasible because attitude is too difficult to recalculate after a processor shutdown.

The report concludes that it was "the decision to shut down the processor operation that finally proved fatal" and recommends that no sensors (such as the inertial reference system) should be allowed to stop sending best effort data. While it is always dangerous to second guess the investigators' conclusions from limited information, it seems that this recommendation should be followed only if the fault (as in this case) does not include critical data. In many cases, sending known bad results may be as bad, or even worse, than reporting the inability to provide good data, i.e., the "best effort" data may be unsafe. At the least, hazard identification and analysis should be performed and realistic hazards eliminated or mitigated in the system design. It should not be assumed that redundancy will eliminate hazards without a thorough analysis of all potential hazard causes, particularly for software-based control systems.

It is unclear why the system engineers would opt to shut down the entire inertial guidance system when an exception was raised in a non-critical function (the alignment function after liftoff). The report suggests:

> The reason behind this drastic action [of shutting down the computer] lies in the culture within the Ariane programme of only addressing random hardware failures. From this point of view, exception—or error—handling mechanisms are designed for a random hardware failure which can quite rationally be handled by a backup system [71].

Mental Model Flaws: This flawed decision might arise from two common mental model flaws:

1. Not understanding the different failure modes of software (design errors versus random failures), i.e., a backup computer running identical software does not provide any protection against software design errors.

2. Assuming that software errors will be eliminated during test and thus no additional protection is needed. This assumption is an extremely common (but obviously erroneous) among engineers.

The Ariane 5 loss was a classic example of a *system accident*: Each component worked as specified but together the specified component behavior led to disastrous system behavior. Not only did each component in isolation work correctly but, in fact, many of the design features that contributed to the accident involved standard recommended practice.

Classical system engineering techniques to protect against accidents caused by component failure usually involve increasing the reliability (integrity) of the individual components, including redundancy to protect against component failure and building margins of error into components to protect against analysis or manufacturing flaws. None of these techniques are useful in protecting against system accidents, where the problems result from system design flaws: Increasing component integrity will have no effect on accidents where the software operated exactly as it was required and designed to do (but the requirements were incorrect). Standard redundancy, while protecting against computer hardware failures, will again not protect against software requirements (or even coding) errors.

The accident report notes that according to the culture of the Ariane program, only random failures are addressed and they are primarily handled with redundancy. Thus there was a bias toward the mitigation of random failure and not system faults caused by design error. The misunderstanding of software failure modes (and how they differ from hardware failure modes) and the assumption that software failures will be like those of hardware are widespread and not just limited to the Ariane program.

5. Inadequate identification of critical components and functions:

Basis for Decision: The report recommends reconsidering the definition used to identify critical components, taking failures of software components into account (particularly single-point failures). But there is no information about what definition was used and what procedures were used to identify critical functions. However, the usual reliance by engineers on FMEA and FTA, which are inadequate for software-intensive systems and for accidents not involving component failure, may be part of the problem. A new hazard analysis technique based on STAMP that is more effective for systms accidents and for software-intensive systems, is described in the next chapter.

6. Decision not to include Ariane 5 data in specifications:

As stated previously, this omission was a choice, not inadvertent. No information is provided in the accident investigation report to explain this decision. Clearly, the environment in which the system will operate and the relevant characteristics of any controlled hardware should be completely and accurately documented and made available to developers, reviewers, and testers.

7. Lack of specification of operational restrictions that emerge from the chosen system design and component implementation:

Operational restrictions on real-time systems, particularly those related to safety, need to be documented and communicated to the software engineers to ensure that the software does not violate these restrictions. In addition, both the system limitations and any additional limitations resulting from implementation decisions need to be considered during test and thus must be passed to the testing and external review processes.

8. Inadequate documentation and communication of design rationale.

The report notes that the structure of the documentation obscured the ability to review the critical design decisions and their underlying rationale. Inadequate documentation of design rationale to allow effective review of design decisions is a very common problem in system and software specifications. Justification for design and coding decisions must be recorded in a way that it can be easily located, reviewed, and kept up to date when changes are made.

8.2.9 Project Management and the Upper Levels of the Development Control Structure

The report is silent about any possible role management and the upper levels of the development structure played in the accident. There is one hint in a final recommendation at the very end of the report that the accident investigators did find a problem:

> "A more transparent organization of the cooperation among partners in the Ariane 5 programme must be considered. Close engineering cooperation, with clear cut authority and responsibility, is needed to achieve system coherence, with simple and clear interfaces between partners."

No additional details are provided.

Was an effective communication and authority structure established? What role did project management play in the reuse decision? Did a system safety program exist and was responsibility, accountability, and authority for safety established? From the information that is provided, there appear to be no controls in place to prevent the system engineering errors from being made or from being detected in time to be correted. There is no indication in the report that hazards were identified or controlled in the conceptual design or implementation of the automated control systems nor that there were safety evaluations and reviews. In fact, there is no mention of a system safety program at all.

Mental Models Flaws: The report does allude to two common mental model flaws among the system engineers and management. First, it appears that an assumption was made that safety and reliability are equivalent for software-based control systems and the focus was on component reliability and correctness (satisfaction of the requirements specification) and on increasing reliability through redundancy. Increasing the reliability of these components, however, would not have helped with this system accident—as is true for most system accidents—as each individual component operated exactly as required (specified).

A second mentla model flaw common to most of the flawed development decisions was underestimating and misunderstanding software-related risks. The accident report says that software was assumed to be correct until it was shown to be faulty. This assumption is widespread in engineering and stems from an underestimation of the complexity of most software and an overestimation of the effectiveness of software testing. Complacency also probably played a part in the inclusion of unnecessary complexity and software functions and many of the other ineffective or inadequate technical activities.

The next section presents an analysis of the Titan/Centaur accident using the new model. More information is included in the Titan/Centaur accident report about the engineering process and how it led to the system design flaws. This additional information provides the opportunity for a better demonstration of the new control-based accident model.

8.3 The Loss of a Milstar Satellite

On April 30, 1999, at 12:30 EDT, a Titan IV B-32 booster equipped with a Centaur TC-14 upper stage was launched from Cape Canaveral. The mission was to place a Milstar-3 satellite into geosynchronous orbit. Milstar is a joint services satellite communications system that provides secure, jam resistant, worldwide communications to meet wartime requirements. It was the most advanced military communications satellite system to that date. The first Milstar satellite was launched February 7, 1994 and the second was launched November 5, 1995. This mission was to be the third launch.

As a result of some anomalous events, the Milstar satellite was placed in an incorrect and unusable low elliptical final orbit, as opposed to the intended geosynchronous orbit. Media interest was high due to this mishap being the third straight Titan IV failure and due to recent failures of other commercial space launches. In addition, this accident is believed to be one of the most costly unmanned losses in the history of Cape Canaveral Launch Operations. The Milstar satellite cost about $800 million and the launcher an additional $433 million.

To their credit, the accident investigation board went beyond the usual chain-of-events model and instead interpreted the accident in terms of a complex and flawed process:

> Failure of the Titan IV B-32 mission is due to a failed software development, testing, and quality assurance process for the Centaur upper stage. That failed process did not detect and correct a human error in the manual entry of the I1(25) roll rate filter constant entered in the Inertial Measurement System flight software file. The value should have been entered as -1.992476, but was entered as -0.1992476. Evidence of the incorrect I1(25) constant appeared during launch processing and the launch countdown, but its impact was not sufficiently recognized or understood and, consequently, not corrected before launch. The incorrect roll rate filter constant zeroed any roll rate data, resulting in the loss of roll axis control, which then caused loss of yaw and pitch control. The loss of attitude control caused excessive firings of the Reaction Control system and subsequent hydrazine depletion. Erratic vehicle flight during the Centaur main engine burns caused the Centaur to achieve an orbit apogee and perigee much lower than desired, which resulted in the Milstar separating in a useless low final orbit [86].

Fully understanding this accident requires understanding why the error in the roll rate filter constant was introduced in the load tape, why it was not found during the load tape production process and internal review processes, why it was not found during the extensive independent verification and validation effort applied to this software, and why it was not detected during operations at the launch site. In other words, why the safety control structure was ineffective in each of these instances.

Figure 8.3 shows the hierarchical control model of the accident, or at least those parts that can be gleaned from the official accident report[4]. Lockheed Martin Astronautics (LMA) was the prime contractor for the mission. The Air Force Space and Missile Systems Center Launch Directorate (SMC) was responsible for insight and administration of the LMA contract. Besides LMA and SMC, the Defense Contract Management Command (DCMC) played an oversight role, but the report is

[4] Some details of the control structure may be incorrect because they were not detailed in the report, but the structure is close enough for the purpose of this chapter.

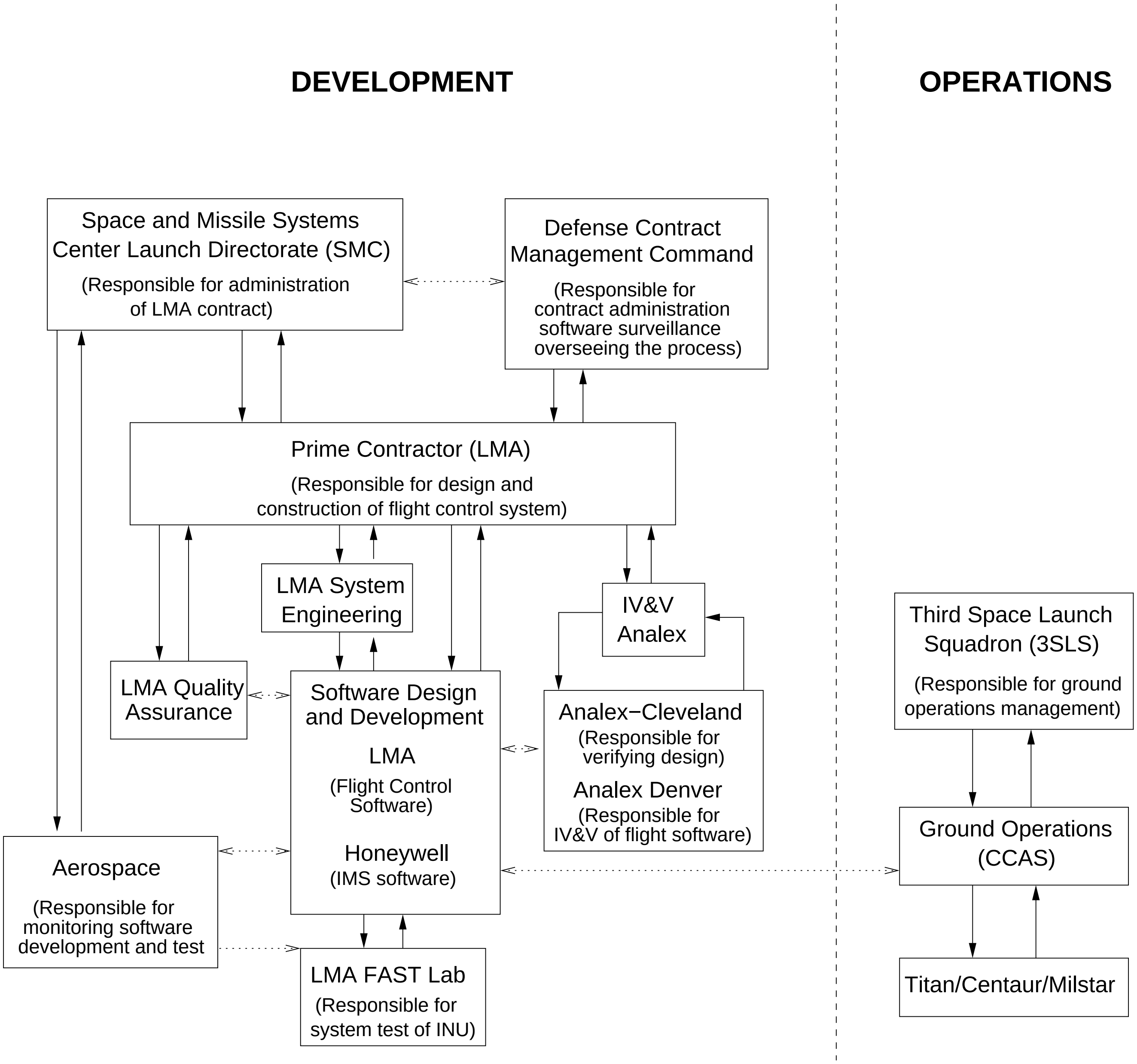

DEVELOPMENT
OPERATIONS
Space and Missile Systems Center Launch Directorate (SMC)
(Responsible for administration of LMA contract)
Defense Contract Management Command
(Responsible for contract administration software surveillance overseeing the process)
Prime Contractor (LMA)
(Responsible for design and construction of flight control system)
LMA System Engineering
IV&V Analex
LMA Quality Assurance
Software Design and Development
LMA
(Flight Control Software)
Honeywell
(IMS software)
Analex−Cleveland
(Responsible for verifying design)
Analex Denver
(Responsible for IV&V of flight software)
Aerospace
(Responsible for monitoring software development and test
LMA FAST Lab
(Responsible for system test of INU)
Third Space Launch Squadron (3SLS)
(Responsible for ground operations management)
Ground Operations (CCAS)
Titan/Centaur/Milstar

Figure 8.3: Hierarchical Control Structure

not clear about what exactly this role was beyond a general statement about responsibility for contract management, software surveillance, and overseeing the development process.

LMA designed and developed the flight control software, while Honeywell was responsible for the IMS software. This separation of control, combined with poor coordination, accounts for some of the problems that occurred. Analex was the independent verification and validation (IV&V) contractor, while Aerospace Corporation provided independent monitoring and evaluation. Ground launch operations at Cape Canaveral Air Station (CCAS) were managed by the Third Space Launch Squadron (3SLS).

Once again, starting from the physical process and working up the levels of control, a STAMP analysis examines each level for the flaws in the process at that level that provided inadequate control of safety in the process level below. The process flaws at each level are then examined and explained in terms of a potential mismatch in models between the controller's model of the process and the real process, incorrect design of the control algorithm, lack of coordination among the control activities, deficiencies in the reference channel, and deficiencies in the feedback or monitoring channel. When human decision-making is involved, the analysis results must also include information about the context in which the decision(s) was made and the information available (and necessary information *not* available) to the decision makers.

One general thing to note in this accident is that there were a large number of redundancies in each part of the process to prevent the loss, but they were not effective. Sometimes (as in this case), built-in redundancy itself causes complacency and overconfidence, as was illustrated by the Ariane 501 accident, and is a critical factor in the accident process. The use of redundancy to provide protection against losses must include a detailed analysis of coverage and any potential gaps in the safety control provided by the redundancy.

8.3.1 The Physical Process (Titan/Centaur/Milstar)

Components of the Physical Process: The Lockheed Martin Astronautics (LMA) Titan IV B is a heavy-lift space launch vehicle used to carry government payloads such as Defense Support Program, Milstar, and National Reconnaissance Office satellites into space. It can carry up to 47,800 pounds into low-earth orbit and up to 12,700 pounds into a geosynchronous orbit. The vehicle can be launched with no upper stage or with one of two optional upper stages, providing greater and varied capability.

The LMA Centaur is a cryogenic, high-energy upper stage. It carries its own guidance, navigation, and control system, which measures the Centaur's position and velocity on a continuing basis throughout flight. It also determines the desired orientation of the vehicle in terms of pitch, yaw, and roll axis vectors. It then issues commands to the required control components to orient the vehicle in the proper attitude and position, using the main engine or the Reaction Control System (RCS) engines (Figure 8.4). The main engines are used to control thrust and velocity. The RCS provides thrust for vehicle pitch, yaw, and roll control, for post-injection separation and orientation maneuvers, and for propellant settling prior to engine restart.

System Hazards: (1) The satellite does not reach a useful geosynchronous orbit; (2) the satellite is damaged during orbit insertion maneuvers and cannot provide its intended function.

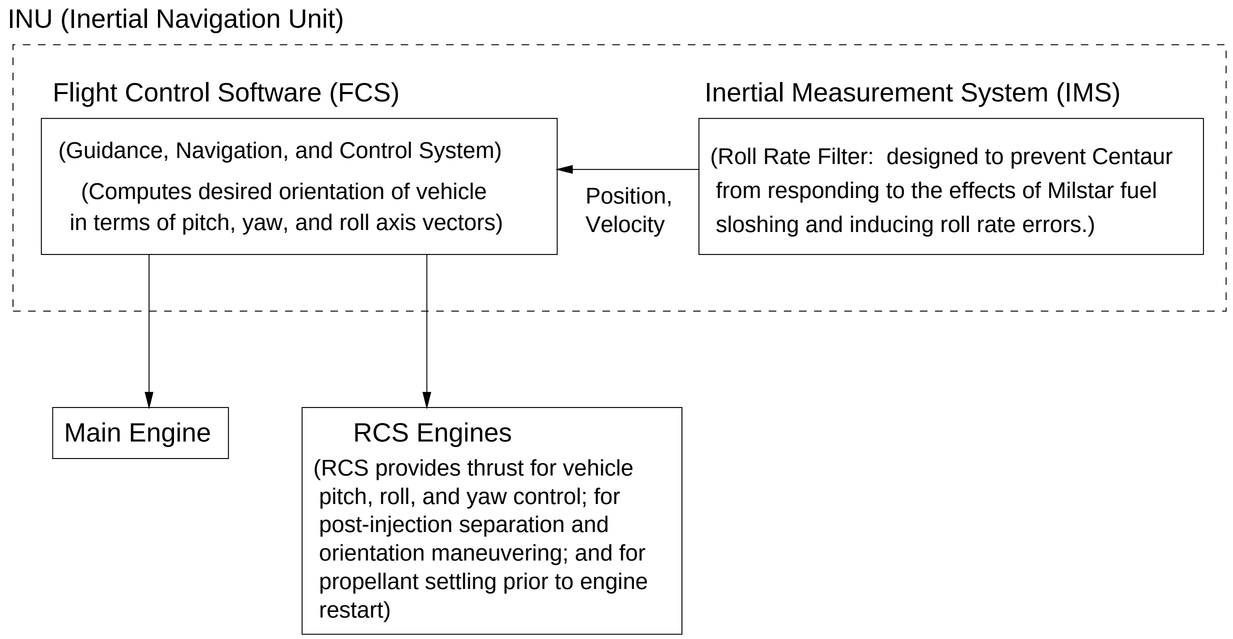

INU (Inertial Navigation Unit)
Flight Control Software (FCS)
(Guidance, Navigation, and Control System)
(Computes desired orientation of vehicle in terms of pitch, yaw, and roll axis vectors)
Inertial Measurement System (IMS)
(Roll Rate Filter: designed to prevent Centaur from responding to the effects of Milstar fuel sloshing and inducing roll rate errors.)
Position, Velocity
Main Engine
RCS Engines
(RCS provides thrust for vehicle pitch, roll, and yaw control; for post-injection separation and orientation maneuvering; and for propellant settling prior to engine restart)

Figure 8.4: Technical Process Control Structure for INU

Description of Process Controller (INU): The Inertial Navigation Unit (INU) has two parts (Figure 8.4): (1) the Guidance, Navigation, and Control System (the Flight Control Software or FCS) and (2) an Inertial Measurement System (IMS). The Flight Control Software computes the desired orientation of the vehicle in terms of the pitch, yaw, and roll axis vectors and issues commands to the main engines and the reaction control system to control vehicle orientation and thrust. To accomplish this goal, the FCS uses position and velocity information provided by the IMS. The component of the IMS involved in the loss is a roll rate filter, which is designed to prevent the Centaur from responding to the effects of Milstar fuel sloshing and thus inducing roll rate errors.

Safety Constraint on FCS that was Violated: The FCS must provide the attitude control, separation, and orientation maneuvering commands to the main engines and the RCS system necessary to attain geosynchronous orbit.

Safety Constraints on IMS that were Violated: The position and velocity values provided to the FCS must not be capable of leading to a hazardous control action. The roll rate filter must prevent the Centaur from responding to the effects of fuel sloshing and inducing roll rate errors.

8.3.2 Description of the Proximal Events Leading to the Loss

There were three planned burns during the Centaur flight. The first burn was intended to put the Centaur into a parking orbit. The second would move the Centaur into an elliptical transfer orbit that was to carry the Centaur and the satellite to geosynchronous orbit. The third and final burn would circularize the Centaur in its intended geosynchronous orbit. A coast phase was planned between each burn. During the coast phase, the Centaur was to progress under its own momentum to the proper point in the orbit for the next burn. The Centaur would also exercise a roll sequence and an attitude control maneuver during the coast periods to provide passive thermal control and to settle the main engine propellants in the bottom of the tanks.

First Burn: The first burn was intended to put the Centaur into a parking orbit. The IMS transmitted a zero or near zero roll rate to the Flight Control software, however, due to the use of an incorrect roll rate filter constant. With no roll rate feedback, the FCS provided inappropriate control commands that caused the Centaur to become unstable about the roll axis and not to roll to the desired first burn orientation. The Centaur began to roll back and forth, eventually creating sloshing of the vehicle liquid fuel in the tanks, which created unpredictable forces on the vehicle and adversely affected flow of fuel to the engines. By the end of the first burn (approximately 11 minutes and 35 seconds after liftoff), the roll oscillation began to affect the pitch and yaw rates of the vehicle as well. The FCS predicted an incorrect time for main engine shutdown due to the effect on the acceleration of the vehicle's tumbling and fuel sloshing. The incorrect shutdown in turn resulted in the Centaur not achieving its intended velocity during the first burn, and the vehicle was placed in an unintended park orbit.

First Coast Phase: During the coast phases, the Centaur was to progress under its own momentum to the proper point in the orbit for the next burn. During this coasting period, the FCS was supposed to command a roll sequence and an attitude control maneuver to provide passive thermal control and to settle the main engine propellants in the bottom of the tanks. Because of the roll instability and transients created by the engine shutdown, the Centaur entered this first coast phase tumbling. The FCS directed the RCS to stabilize the vehicle. Late in the park orbit, the Centaur was finally stablized about the pitch and yaw axes, although it continued to oscillate about the roll axis. In stabilizing the vehicle, however, the RCS expended almost 85 percent of the RCS system propellant (hydrazine).

Second Burn: The FCS successfully commanded the vehicle into the proper attitude for the second burn, which was to put the Centaur and the satellite into an elliptical transfer orbit that would carry them to geosynchronous orbit. The FCS ignited the main engines at approximately one hour, six minutes, and twenty-eight seconds after liftoff. Soon after entering the second burn phase, however, inadequate FCS control commands caused the vehicle to again become unstable about the roll axis and begin a diverging roll oscillation.

Because the second burn is longer than the first, the excess roll commands from the FCS eventually saturated the pitch and yaw channels. At approximately two minutes into the second burn, pitch and yaw control was lost (as well as roll), causing the vehicle to tumble for the remainder of the burn. Due to its uncontrolled tumbling during the burn, the vehicle did not achieve the planned acceleration for transfer orbit.

Second Coast Phase (transfer orbit): The RCS attempted to stabilize the vehicle but it continued to tumble. The RCS depleted its remaining propellant approximately twelve minutes after the FCS shut down the second burn.

Third Burn: The goal of the third burn was to circularize the Centaur in its intended geosynchronous orbit. The FCS started the third burn at two hours, thirty-four minutes, and fifteen seconds after liftoff. It was started earlier and was shorter than had been planned. The vehicle tumbled throughout the third burn, but without the RCS there was no way to control it. Space vehicle separation was commanded at approximately two hours after the third burn began, resulting in the Milstar being placed in a useless low elliptical orbit, as opposed to the desired geosynchronous orbit (Figure 8.5).

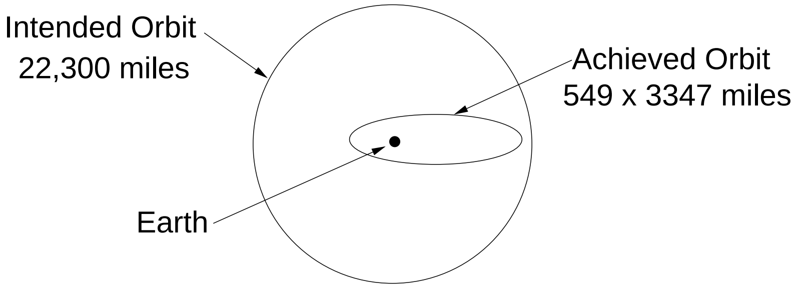


Figure 8.5: Achieved Orbit vs. Intended Orbit

Post Separation: The Mission Director ordered early turn-on of the satellite in an attempt to save it, but the ground controllers were unable to contact the satellite for approximately three hours. Six hours and fourteen minutes after liftoff, control was acquired and various survival and emergency actions were taken. The satellite had been damaged from the uncontrolled vehicle pitch, yaw, and roll movements, however, and there were no possible actions the ground controllers could have taken in response to the anomalous events that would have saved the mission.

The mission was officially declared a failure on May 4, 1999, but personnel from LMA and the Air Force controlled the satellite for six additional days in order to place the satellite in a non-interfering orbit with minimum risk to operational satellites. It appears the satellite performed as designed, despite the anomalous conditions. It was shut down by ground control on May 10, 1999.

8.3.3 Physical Process and Automated Controller Failures and Dysfunctional Interactions

Figure 8.6 shows the automated controller flaws leading to the accident. The Inertial Measurement System algorithm was incorrect; specifically, there was an incorrect roll rate filter constant in the IMS software file (Figure 8.6) that led to a dysfunctional interaction with the flight control software. However, the algorithm operated as designed (i.e., it did not fail).

The Flight Control Software operated correctly (i.e., according to its requirements). However, it received incorrect input from the IMS, leading to an incorrect internal FCS software model of the process—the roll rate was thought to be zero or near zero when it was not. Thus there was a mismatch between the FCS internal model of the process state and the real process state. This mismatch led to the RCS issuing incorrect control commands to the main engine (to shut down early) and to the RCS engines. Using STAMP terminology, the loss resulted from a dysfunctional interaction between the FCS and the IMS. Neither failed—they operated correctly with respect to the instructions (including constants) and data provided.

The accident report does not explore whether the FCS software could have included sanity checks on the roll rate or vehicle behavior to detect that incorrect roll rates were being provided by the IMS. Even if the FCS did detect it was getting anomalous roll rates, there may not have been any recovery or fail-safe behavior that could have been designed into the system. Without more information about the Centaur control requirements and design, it is not possible to speculate about whether the Inertial Navigation Unit software (the IMS and FCS) might have been designed to be fault tolerant with respect to filter constant errors.

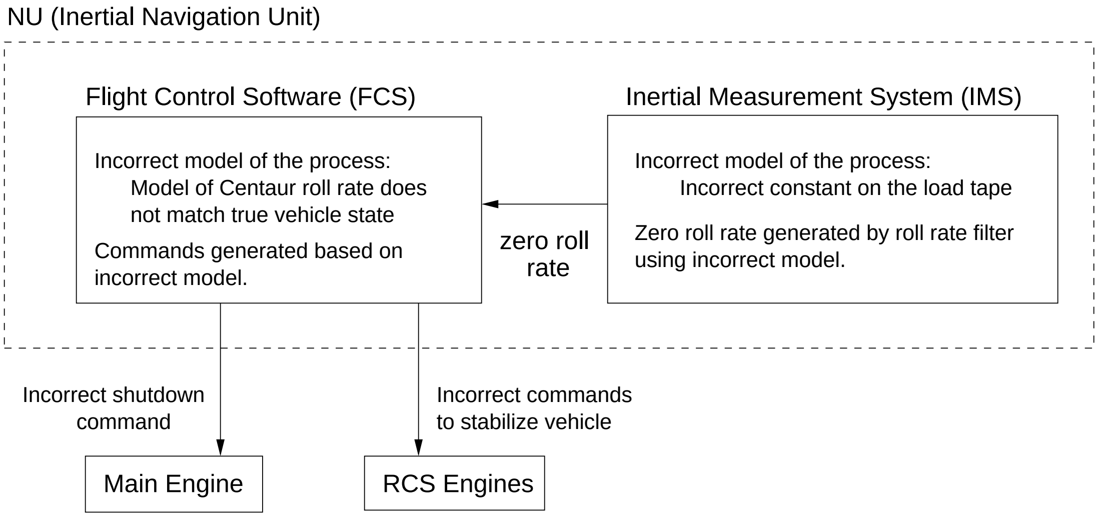

NU (Inertial Navigation Unit)
Flight Control Software (FCS)
Incorrect model of the process:
Model of Centaur roll rate does not match true vehicle state
Commands generated based on incorrect model.
Inertial Measurement System (IMS)
Incorrect model of the process:
Incorrect constant on the load tape
Zero roll rate generated by roll rate filter using incorrect model.
zero roll rate
Incorrect shutdown command
Incorrect commands to stabilize vehicle
Main Engine
RCS Engines

Figure 8.6: Control Flaws at the Physical Process and Software Controller Levels

Process Models: Both the FCS and the IMS had process models that did not match the real process state, leading to the hazardous outputs from the software. The FCS model of the vehicle orientation did not match the actual orientation due to incorrect input about the state of a controlled variable (the roll rate). The IMS provided the bad input because of an incorrect model of the process, i.e., the I1(25) constant used in the roll rate filter, i.e., the feedback or monitoring channel of the FCS provided incorrect feedback about the roll rate.

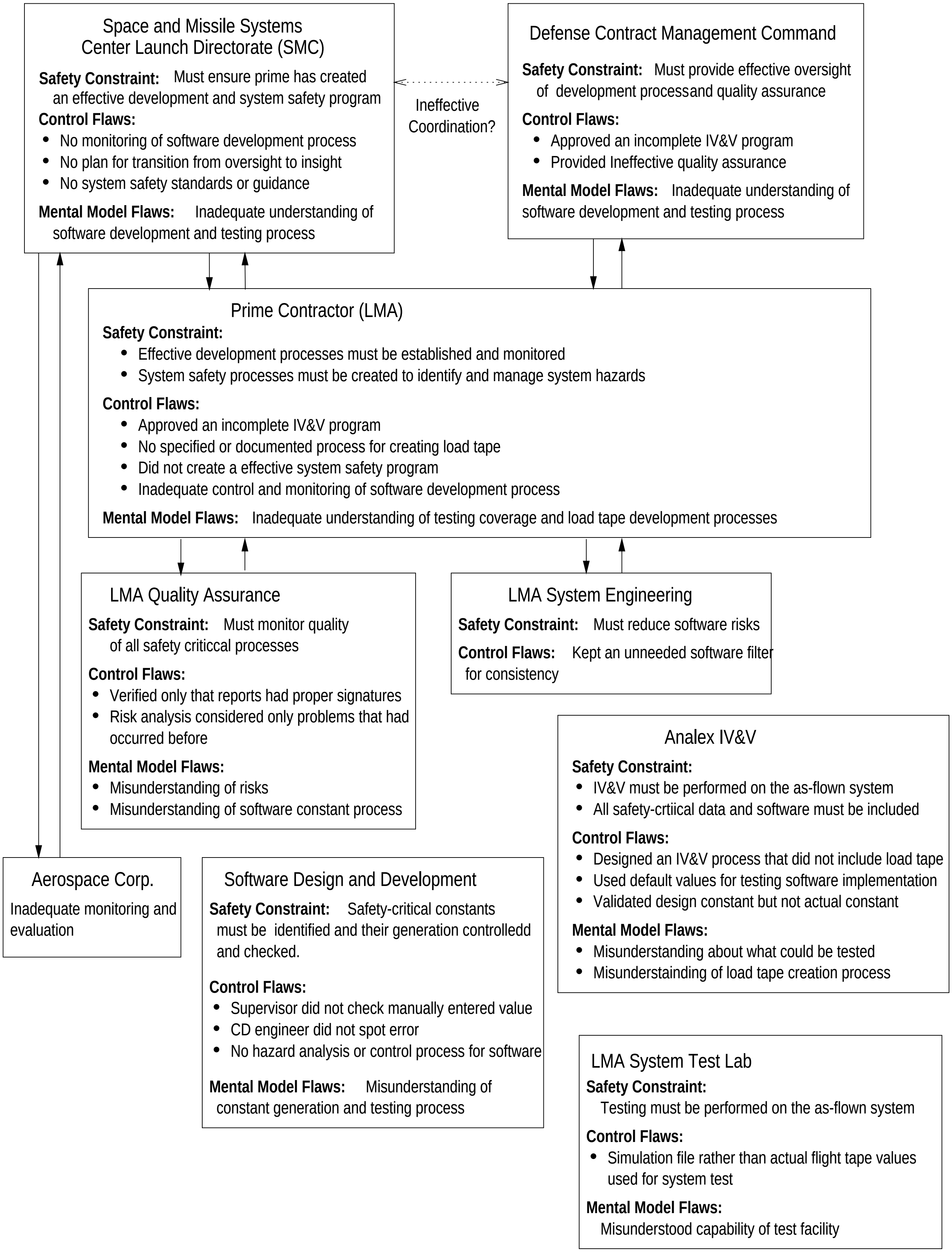
This level of explanation of the flaws in the process (the vehicle and its flight behavior) as well as its immediate controller provides a description of the "symptom," but does not provide enough information about the factors involved to prevent reoccurrences. Simply fixing that particular flight tape is not enough. We need to look at the higher levels of the control structure for that. Several questions need to be answered before the accident can be understood: Why was the roll rate error not detected during launch operations? Why was an erroneous load tape created in the first place? Why was the error not detected in the regular verification and validation process or during the extensive independent verification and validation process? How did the error get past the quality assurance process? What role did program management play in the accident? This accident report does a much better job in answering these types of questions than the Ariane 5 report.

Figures 8.7 and 8.8 summarize the information in the rest of this section.

8.3.4 Launch Site Operations

The function of launch site operations is to monitor launch pad behavior and tests and to detect any critical anomalies prior to flight. Why was the roll rate error not detected during launch operations?

Process Being Controlled: Preparations for launch at the launch site as well as the launch itself.



Space and Missile Systems Center Launch Directorate (SMC)

Safety Constraint: Must ensure prime has created an effective development and system safety program

Control Flaws:

- No monitoring of software development process
- No plan for transition from oversight to insight
- No system safety standards or guidance

Mental Model Flaws: Inadequate understanding of software development and testing process

Defense Contract Management Command

Safety Constraint: Must provide effective oversight of development processand quality assurance

Control Flaws:

- Approved an incomplete IV&V program
- Provided Ineffective quality assurance

Mental Model Flaws: Inadequate understanding of software development and testing process

Prime Contractor (LMA)

Safety Constraint:

- Effective development processes must be established and monitored
- System safety processes must be created to identify and manage system hazards

Control Flaws:

- Approved an incomplete IV&V program
- No specified or documented process for creating load tape
- Did not create a effective system safety program
- Inadequate control and monitoring of software development process

Mental Model Flaws: Inadequate understanding of testing coverage and load tape development processes

LMA Quality Assurance

Safety Constraint: Must monitor quality of all safety criticcal processes

Control Flaws:

- Verified only that reports had proper signatures
- Risk analysis considered only problems that had occurred before

Mental Model Flaws:

- Misunderstanding of risks
- Misunderstanding of software constant process

LMA System Engineering

Safety Constraint: Must reduce software risks

Control Flaws: Kept an unneeded software filter for consistency

Analex IV&V

Safety Constraint:

- IV&V must be performed on the as-flown system
- All safety-crtiical data and software must be included

Control Flaws:

- Designed an IV&V process that did not include load tape
- Used default values for testing software implementation
- Validated design constant but not actual constant

Mental Model Flaws:

- Misunderstanding about what could be tested
- Misunderstainding of load tape creation process

Aerospace Corp.

Inadequate monitoring and evaluation

Software Design and Development

Safety Constraint: Safety-critical constants must be identified and their generation controlledd and checked.

Control Flaws:

- Supervisor did not check manually entered value
- CD engineer did not spot error
- No hazard analysis or control process for software

Mental Model Flaws: Misunderstanding of constant generation and testing process

LMA System Test Lab

Safety Constraint: Testing must be performed on the as-flown system

Control Flaws:

- Simulation file rather than actual flight tape values used for system test

Mental Model Flaws: Misunderstood capability of test facility

Figure 8.7: STAMP model of Development Process

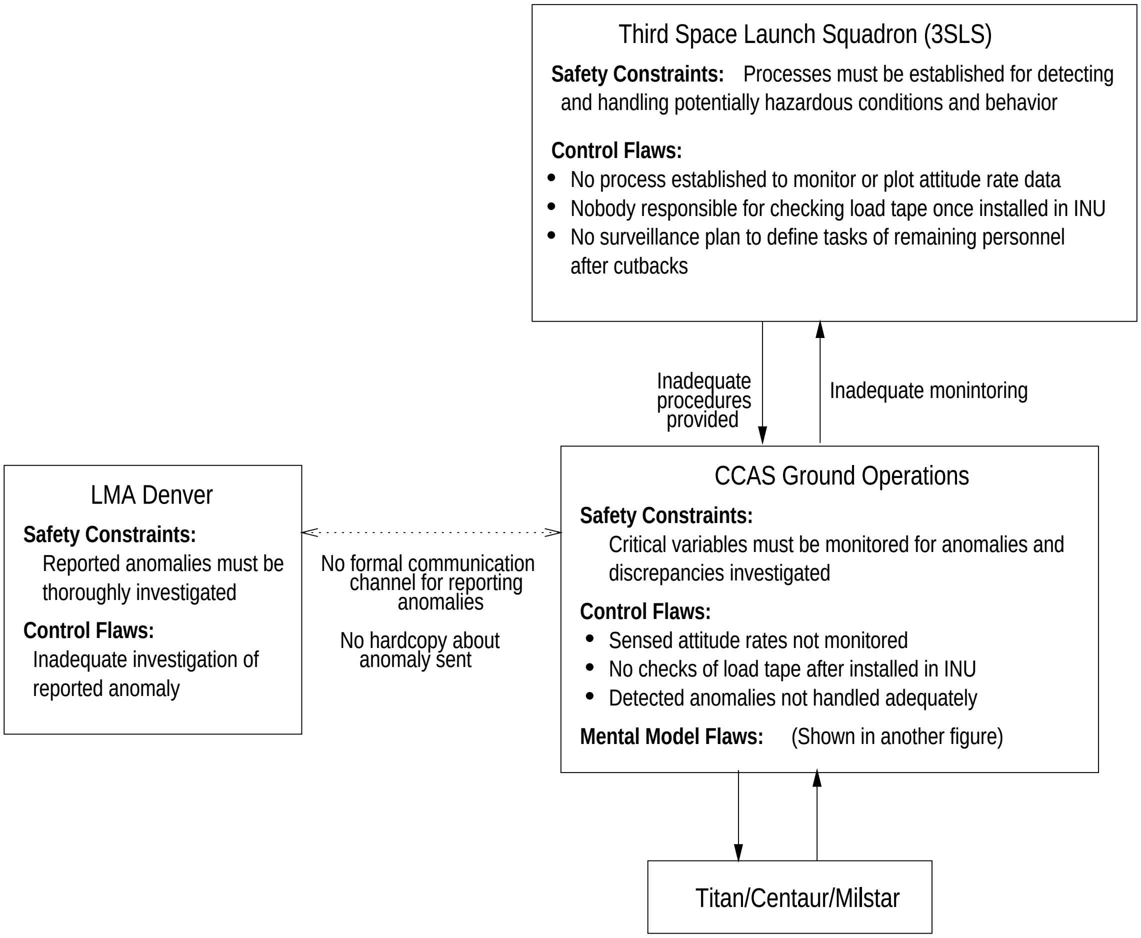

Third Space Launch Squadron (3SLS)
Safety Constraints: Processes must be established for detecting and handling potentially hazardous conditions and behavior
Control Flaws:
• No process established to monitor or plot attitude rate data
• Nobody responsible for checking load tape once installed in INU
• No surveillance plan to define tasks of remaining personnel after cutbacks
Inadequate procedures provided
Inadequate monintoring
LMA Denver
Safety Constraints:
Reported anomalies must be thoroughly investigated
Control Flaws:
Inadequate investigation of reported anomaly
No formal communication channel for reporting anomalies
No hardcopy about anomaly sent
CCAS Ground Operations
Safety Constraints:
Critical variables must be monitored for anomalies and discrepancies investigated
Control Flaws:
• Sensed attitude rates not monitored
• No checks of load tape after installed in INU
• Detected anomalies not handled adequately
Mental Model Flaws: (Shown in another figure)
Titan/Centaur/Milstar

Figure 8.8: STAMP model of Launch Operations Process

Safety Constraint Violated: Critical variables (including those in software) must be monitored and errors detected before launch. Potentially hazardous anomalies detected at the launch site must be formally logged and thoroughly investigated and handled.

Context: Management had greatly reduced the number of engineers working launch operations, and those remaining were provided with few guidelines as to how they should perform their job. The accident report says that their tasks were not defined by their management so they used their best engineering judgment to determine which tasks they should perform, which variables they should monitor, and how closely to analyze the data associated with each of their monitoring tasks.

Controls: The controls are not described well in the report. From what is included, it does not appear that controls were implemented to monitor or detect software errors at the launch site although a large number of vehicle variables were monitored.

Roles and Responsibilities: The report is also not explicit about the roles and responsibilities of those involved. LMA had launch personnel at CCAS, including Product Integrity Engineers (PIEs). 3SLS had launch personnel to control the launch process as well as software to check process variables and to assist the operators in evaluating observed data.

Failures, Dysfunctional Interactions, Flawed Decisions, and Inadequate Control Actions: Despite clear indications of a problem with the roll rate information being produced by the IMS, it was not detected by some launch personnel who should have and detected but mishandled by others. Specifically:

1. One week before launch, LMA personnel at CCAS observed much lower roll rate filter values than they expected. When they could not explain the differences at their level, they raised their concerns to Denver LMA Guidance Product Integrity Engineers (PIEs), who were now at CCAS. The on-site PIEs could not explain the differences either, so they directed the CCAS personnel to call the control dynamics (CD) design engineers in Denver. On Friday, April 23, the LMA Guidance Engineer telephoned the LMA CD lead. The CD lead was not in his office so the Guidance Engineer left a voice mail stating she noticed a significant change in roll rate when the latest filter rate coefficients were entered. She requested a return call to her or to her supervisor. The Guidance Engineer also left an email for her supervisor at CCAS explaining the situation. Her supervisor was on vacation and was due back at the office Monday morning April 26, when the Guidance Engineer was scheduled to work the second shift. The CD lead and the CD engineer who originally specified the filter values listened to the voice mail from the Guidance Engineer. They called her supervisor at CCAS who had just returned from vacation. He was initially unable to find the email during their conversation. He said he would call back, so the CD engineer left the CD lead's office. The CD lead subsequently talked to the Guidance Engineer's supervisor after he found and read the email. The CD lead told the supervisor at CCAS that the filter values had changed in the flight tape originally loaded on April 14, 1999, and the roll rate output should also be expected to change. Both parties believed the difference in roll rates observed were attributable to expected changes with the delivery of the flight tape.

2. On the day of the launch, a 3SLS INU Product Integrity Engineer (PIE) at CCAS noticed the low roll rates and performed a rate check to see if the gyros were operating properly. Unfortunately, the programmed rate check used a default set of I1 constants to filter the measured rate and consequently reported that the gyros were sensing the earth rate correctly. If the sensed attitude rates had been monitored at that time or if they had been summed and plotted to ensure they were properly sensing the earth's gravitational rate, the roll rate problem could have been identified.

3. A 3SLS engineer also saw the roll rate data at the time of tower rollback, but was not able to identify the problem with the low roll rate. He had no documented requirement or procedures to review the data and no reference to compare to the roll rate actually being produced.

The communication channel between LMA Denver and the LMA engineers at CCAS was clearly flawed. The accident report provides no information about any established reporting channel from the LMA CCAS or LMA Denver engineers to a safety organization or up the management chain. No "alarm" system adequate to detect the problem or that it was not being adequately handled seems to have existed. The report says there was confusion and uncertainty from the time the roll rate anomaly was first raised by the CCAS LMA engineer in email and voice mail until it was "resolved" as to how it should be reported, analyzed, documented, and tracked because it was a "concern" and not a "deviation." There is no explanation of these terms nor any description of a formal problem reporting and handling system in the accident report.

Inadequate Control Algorithm: The accident report says that at this point in the prelaunch process, there was no process to monitor or plot attitude rate data, that is, to perform a check to see if the attitude filters were properly sensing the earth's rotation rate. Nobody was responsible for checking the load tape constants once the tape was installed in the INU at the launch site. Therefore, nobody was able to question the anomalous rate data recorded or correlate it to the low roll rates observed about a week prior to launch and on the day of launch. In addition, the LMA engineers at Denver never asked to see a hard copy of the actual data observed at CCAS, nor did they talk to the guidance engineer or Data Station Monitor at CCAS who questioned the low filter rates. They simply explained it away as attributable to expected changes associated with the delivery of the flight tape.

Process Model Flaws: Five models are involved here (see Figure 8.9):

1. Ground rate check software: The software used to do a rate check on the day of launch used default constants instead of the actual load tape. Thus there was a mismatch between the model used in the ground rate checking software and the model used by the actual IMS software.

2. Ground crew models of the development process: Although the report does not delve into this factor, it is very possible that complacency may have been involved and that the model of the thoroughness of the internal quality assurance and external IV&V development process in the minds of the ground operations personnel as well as the LMA guidance engineers who were informed of the observed anomalies right before launch did not match the real development process. There seemed to be no checking of the correctness of the software after the standard

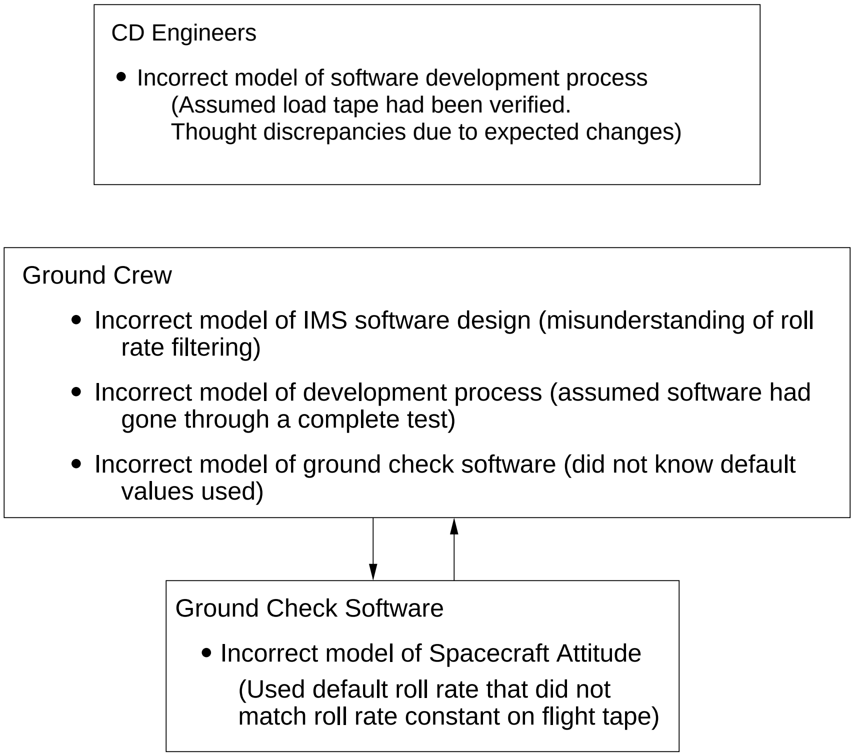

CD Engineers
• Incorrect model of software development process
(Assumed load tape had been verified.
Thought discrepancies due to expected changes)
Ground Crew
• Incorrect model of IMS software design (misunderstanding of roll rate filtering)
• Incorrect model of development process (assumed software had gone through a complete test)
• Incorrect model of ground check software (did not know default values used)
Ground Check Software
• Incorrect model of Spacecraft Attitude
(Used default roll rate that did not
match roll rate constant on flight tape)

Figure 8.9: The Flawed Process Models used by the Ground Personnel and Software

testing during development. Hardware failures are usually checked up to launch time, but often testing is assumed to have removed all software errors and therefore further checks are not needed.

3. Ground crew models of the IMS software design: The ground launch crew had an inadequate understanding of how the roll rate filters worked. No one other than the control dynamics engineers who designed the I1 roll rate constants understood their use or the impact of filtering the roll rate to zero. So when discrepancies were found before launch, nobody at the launch site understood the I1 roll rate filter design well enough to detect the error.

4. Ground crew models of the rate check software: Apparently, the ground crew was unaware that the checking software used default values for the filter constants.

5. CD engineers' model of the flight tape change: The control dynamics lead engineer at the launch site and her supervisor at LMA Denver thought that the roll rate anomalies were due to known changes in the flight tape. Neither went back to the engineers themselves to check this conclusion with those most expert in the details of the Centaur control dynamics.

Coordination: Despite several different groups being active at the launch site, nobody had been assigned responsibility for monitoring the software behavior after it was loaded into the INU. The accident report does not mention coordination problems, although it does say there was a lack of understanding of each other's responsibilities between the LMA launch personnel (at CCAS) and the development personnel at LMA Denver and that this led to the concerns of the LMA personnel at CCAS not being adequately addressed.

A more general question that might have been investigated was whether the failure to act properly after detecting the roll rate problem involved a lack of coordination and communication problems between LMA engineers at CCAS and 3SLS personnel. Why did several people notice the problem with the roll rate but do nothing and why were the anomalies they noticed not effectively communicated to those who could do something about it? Several types of coordination problems might have existed. For example, there might have been an overlap problem, with each person who saw the problem assuming that someone else was handling it or the problem might have occurred at the boundary between several people's responsibilities.

Feedback: There was a missing or inadequate feedback channel from the launch personnel to the development organization.

Tests right before launch detected the zero roll rate, but there was no formal communication channel established for getting that information to those who could understand it. Instead voice mail and email were used. The report is not clear, but either there was no formal anomaly reporting and tracking system or it was not known or used by the process participants.[5]

The LMA (Denver) engineers requested no hardcopy information about the reported anomaly and did not speak directly with the Guidance engineer or Data Station Monitor at CCAS.

[5] Several recent aerospace accidents have involved the bypassing of formal anomaly reporting channels and the substitution of informal email and other communication—with similar results (see Appendix B).

8.3.5 Air Force Launch Operations Management: Third Space Launch Squadron (3SLS)

Process Being Controlled: Activities of the CCAS personnel at the launch site (ground launch operations).

Safety Constraint: Processes must be established for detecting and handling potentially hazardous conditions and behavior detected during launch preparations.

Context: 3SLS management was transitioning from an *oversight* role to an *insight* one without a clear definition of what such a transition might mean or require.

Control Algorithm Flaws: After the ground launch personnel cutbacks, 3SLS management did not create a master surveillance plan to define the tasks of the remaining personnel (the formal insight plan was still in draft). In particular, there were no formal processes established to check the validity of the I1 filter constants or to monitor attitude rates once the flight tape was loaded into the INU at Cape Canaveral Air Station (CCAS) prior to launch. 3SLS launch personnel were provided with no documented requirement nor procedures to review the data and no references with which to compare the observed data in order to detect anomalies.

Process Model: It is possible that misunderstandings (an incorrect model) about the thoroughness of the development process led to a failure to provide requirements and processes for performing software checks at the launch site. Complacency may also have been involved, i.e., the common assumption that software does not fail and that software testing is exhaustive and therefore additional software checking was not needed. However, this is speculation as the report does not explain why management did not provide documented requirements and procedures to review the launch data nor ensure the availability of references for comparison so that discrepancies could be discovered.

Coordination: The lack of oversight led to a process that did not assign anyone the responsibility for some specific launch site tasks.

Feedback or Monitoring Channel: Apparently, launch operations management had no "insight" plan in place to monitor the performance of the launch operations process. There is no information included in the accident report about the process to monitor the performance of the launch operations process or what type of feedback was used (if any) to provide insight into the process.

8.3.6 Software/System Development of the Centaur Flight Control System

Too often, accident investigators stop at this point after identifying operational errors that, if they had not occurred, might have prevented the loss. Occasionally operations management is faulted. Operator errors provide a convenient place to stop in the backward chain of events from the loss event. To their credit, the accident investigation board in this case kept digging. To understand why an erroneous flight tape was created in the first place (and to learn how to prevent a similar occurrence in the future), the software and system development process associated with generating the tape needs to be examined.

Process Description: The INU consists of two major software components developed by different companies: LMA developed the Flight Control System software and was responsible for overall INU testing while Honeywell developed the IMS and was partially responsible for its software development and testing. The I1 constants are processed by the Honeywell IMS, but were designed and tested by LMA.

Safety Constraint Violated: Safety-critical constants must be identified and their generation controlled and checked.

Dysfunctional Interactions, Flawed Decisions, and Inadequate Control Actions: A software Constants and Code Words memo was generated by the LMA Control Dynamics (CD) group and sent to the LMA Centaur Flight Software (FS) group on December 23, 1997. It provided the intended and correct values for the first I1 constants in hardcopy form. The memo also allocated space for 10 additional constants to be provided by the LMA Avionics group at a later time and specified a path and file name for an electronic version of the first 30 constants. The memo did not specify or direct the use of either the hardcopy or the electronic version for creating the constants database.

In early February, 1999, the LMA Centaur FS group responsible for accumulating all the software and constants for the flight load tape was given discretion in choosing a baseline data file. The flight software engineer who created the database dealt with over 700 flight constants generated by multiple sources, in differing formats, and at varying time (some with multiple iterations) all of which had to be merged into a single database. Some constant values came from electronic files that could be merged into the database, while others came from paper memos manually input into the database.

When the FS engineer tried to access the electronic file specified in the software Constants and Code Words Memo, he found the file no longer existed at the specified location on the electronic file folder because it was now over a year after the file had been originally generated. The FS engineer selected a different file as a baseline that only required him to change five I1 values for the digital roll rate filter (an algorithm with five constants). The filter was designed to prevent the Centaur from responding to the effects of Milstar fuel sloshing and inducing roll rate errors at 4 radians/second. During manual entry of those five I1 roll rate filter values, the LMA FS engineer incorrectly entered or missed the exponent for the I1(25) constant. The correct value of the I1(25) filter constant was -1.992476. The exponent should have been a one but instead was entered as a zero, making the entered constant one tenth of the intended value or -0.1992476. The flight software engineer's immediate supervisor did not check the manually entered values.

The only person who checked the manually input I1 filter rate values, besides the flight software engineer who actually input the data, was an LMA Control Dynamics engineer. The FS engineer who developed the Flight Load tape notified the CD engineer responsible for design of the first thirty I1 constants that the tape was completed and the printout of the constants was ready for inspection. The CD engineer went to the FS offices and looked at the hardcopy listing to perform the check and sign off the I1 constants. The manual and visual check consisted of comparing a list of I1 constants from Appendix C of the Software Constants and Code Words Memo to the paper printout from the Flight Load tape. The formats of the floating-point numbers (the decimal and exponent formats) were different on each of these paper documents for the three values cross-checked for each I1 constant. The CD engineer did not spot the exponent error for I1(25) and signed off that the I1 constants on the Flight Load tape were correct. He did not know that the design values had been inserted manually into the database used to build the flight tapes (remember, the values had been stored electronically but the original database no longer existed) and that they were never formally tested in any simulation prior to launch.

The CD engineer's immediate supervisor, the lead for the CD section, did not review the Signoff Report nor catch the error. Once the incorrect filter constant went undetected in the Signoff Report, there were no other formal checks in the process to ensure the I1 filter rate values used in flight matched the designed filter.

Control Algorithm Flaws:

- A process input was missing (the electronic file specified in the Software Constants and Code Words memo), so an engineer regenerated it, making a mistake in doing so.
- Inadequate control was exercised over the constants process. No specified or documented software process existed for electronically merging all the inputs into a single file. There was also no formal, documented process to check or verify the work of the flight software engineer in creating the file. Procedures for creating and updating the database were left up to the flight software engineer's discretion.
- Once the incorrect filter constant went undetected in the Signoff Report, there were no other formal checks in the process to ensure the I1 filter rate values used in flight matched the designed filter.
- The hazard analysis process was inadequate, and no control was exercised over the potential hazard of manually entering incorrect constants, a very common human error. If system safety engineers had identified the constants as critical, then a process would have existed for monitoring the generation of these critical variables. In fact, neither the existence of a system safety program nor any form of hazard analysis are mentioned in the accident report. If such a program had existed, one would think it would be mentioned.

  The report does say that Quality Assurance engineers performed a risk analysis, but they considered only those problems that had happened before:

  > Their risk analysis was not based on determining steps critical to mission success, but on how often problems previously surfaced in particular areas on past launches. They determined software constant generation was low risk because there had not been previous problems in that area. They only verified that the signoff report containing the constants had all the proper signatures[86].

  Considering only the causes of past accidents is not going to be effective for software problems or when new technology is introduced into a system. Computers are, in fact, introduced in order to make previously infeasible changes in functionality and design, which reduces the effectiveness of a "fly-fix-fly" approach to safety engineering. Proper hazard analyses examining all the ways the system components can contribute to an accident need to be performed.

Process Model Flaws: The accident report suggests that many of the various partners were confused about what the other groups were doing. The LMA software personnel who were responsible for creating the database (from which the flight tapes are generated) were not aware that IV&V testing did not use the as-flown (manually input) I1 filter constants in their verification and validation process. The LMA Control Dynamics engineer who designed the I1 rate filter also did not know that the design values were manually input into the database used to build the flight tapes and that the values were never formally tested in any simulation prior to launch.

While the failure of the LMA CD engineer who designed the I1 rate filter to find the error during his visual check was clearly related to the difficulty of checking long lists of differently formatted numbers, it also may have been partly due to less care being taken in the process due to an incorrect mental model, i.e., (1) he did not know the values were manually entered into the database (and were not from the electronic file he had created), (2) he did not know the load tape was never formally tested in any simulation prior to launch, and (3) he was unaware the load tape constants were not used in the IV&V process.

Coordination: The fragmentation/stovepiping in the flight software development process, coupled with the lack of comprehensive and defined system and safety engineering processes, resulted in poor and inadequate communication and coordination among the many partners and subprocesses.

Because the IMS software was developed by Honeywell, most everyone (LMA control dynamics engineers, flight software engineers, product integrity engineers, SQA, IV&V, and DCMC personnel) focused on the FCS and had little knowledge of the IMS software.

8.3.7 Quality Assurance (QA)

Process Being Controlled: The quality of the guidance, navigation, and control system design and development.

Safety Constraint: QA must monitor the quality of all safety-critical processes.

Process Flaw: The internal LMA quality assurance processes did not detect the error in the role rate filter constant software file.

Control Algorithm Flaws: QA verified only that the signoff report containing the load tape constants had all the proper signatures, an obviously inadequate process. This accident is indicative of the problems with QA as generally practiced and why it is often ineffective. The LMA Quality Assurance Plan used was a top-level document that focused on verification of process completion, not on how the processes were executed or implemented. It was based on the original General Dynamics Quality Assurance Plan with recent updates to ensure compliance with ISO 9001. According to this plan, the LMA Software Quality Assurance staff was required only to verify that the signoff report containing the constants had all the proper signatures; they left the I1 constant generation and validation process to the flight software and control dynamics engineers. Software Quality Assurance involvement was limited to verification of software checksums and placing quality assurance stamps on the software products that were produced.

8.3.8 Developer Testing Process

Once the error was introduced into the load tape, it could potentially have been detected during verification and validation. Why did the very comprehensive and thorough developer and independent verification and validation process miss this error?

Safety Constraint Violated: Testing must be performed on the as-flown software (including load tape constants).

Flaws in the Testing Process: The INU (FCS and IMS) was never tested using the actual constants on the load tape:

- Honeywell wrote and tested the IMS software, but they did not have the actual load tape.
- The LMA Flight Analogous Simulation Test (FAST) lab was responsible for system test, i.e., they tested the compatibility and functionality of the flight control software and the Honeywell IMS. But the FAST lab testing used a 300 Hertz filter simulation data file for IMS filters and not the flight tape values. The simulation data file was built from the original, correctly specified values of the designed constants (specified by the LMA CS engineer), not those entered by the software personnel in the generation of the flight load tape. Thus the mix of actual flight software and simulated filters used in the FAST testing did not contain the I1(25) error, and the error could not be detected by the internal LMA testing.

Process Model Mismatch: The testing capability that the current personnel thought the lab had did not match the real capability. The LMA FAST facility was used predominantly to test flight control software developed by LMA. The lab had been originally constructed with the capability to exercise the actual flight values for the I1 roll rate filter constants, but that capability was not widely known by the current FAST software engineers until after this accident; knowledge of the capability had been lost in the corporate consolidation/evolution process so the current software engineers used a set of default roll rate filter constants. Later it was determined that had they used the actual flight values in their simulations prior to launch, they would have caught the error.

8.3.9 Independent Verification and Validation (IV&V)

Safety Constraint Violated: IV&V must be performed on the as-flown software and constants. All safety-critical data and software must be included in the IV&V process.

Dysfunctional Interactions: Each component of the IV&V process performed its function correctly, but the overall design of the process was flawed. In fact, it was designed in such a way that it was not capable of detecting the error in the role rate filter constant.

Analex was responsible for the overall IV&V effort of the flight software. In addition to designing the IV&V process, Analex-Denver performed the IV&V of the flight software to ensure the autopilot design was properly implemented in the software while Analex-Cleveland verified the design of the autopilot but not its implementation. The “truth baseline” provided by LMA, per agreement between LMA and Analex, was generated from the constants verified in the Signoff Report.

In testing the flight software implementation, Analex-Denver used IMS default values instead of the actual I1 constants contained on the flight tape. Generic or default I1 constants were used because they believed the actual I1 constants could not be adequately validated in their rigid body simulations, i.e., the rigid body simulation of the vehicle would not exercise the filters sufficiently[6]. They found out after the mission failure that had they used the actual I1 constants in their simulation, they would have found the order of magnitude error.

Analex-Denver also performed a range check of the program constants and the Class I flight constants and verified that format conversions were done correctly. However the process did not require Analex-Denver to check the accuracy of the numbers in the truth baseline, only to do a range check and a bit-to-bit comparison against the firing tables, which contained the wrong constant. Thus the format conversions they performed simply compared the incorrect I1(25) value in the firing tables to the incorrect I1(25) value after the conversion, and they matched. They did not verify that the designed I1 filter constants were the ones actually used on the flight tape.

Analex-Cleveland had responsibility for verifying the functionality of the design constant but not the actual constant loaded into the Centaur for flight. That is, they were validating the design only and not the "implementation" of the design. Analex-Cleveland received the Flight Dynamics and Control Analysis Report (FDACAR) containing the correct value for the roll filter constant. Their function was to validate the autopilot design values provided in the FDACAR. That does not include IV&V of the I1 constants in the flight format. The original design work was correctly represented by the constants in the FDACAR. In other words, the filter constant in question was listed in the FDACAR with its correct value of -1.992476, and not the value on the flight tape (-0.1992476).

Control Algorithm Flaws: Analex developed (with LMA and government approval) an IV&V program that did not verify or validate the I1 filter rate constants actually used in flight. The I1 constants file was not sent to Analex-Cleveland for autopilot validation because Analex-Cleveland only performed design validation. Analex-Denver used default values for testing and never validated the actual I1 constants used in flight.

Process Model Mismatches: The decision to use default values for testing (both by LMA FAST lab and by Analex-Denver) was based on a misunderstanding about the development and test environment and what was capable of being tested. Both the LMA FAST lab and Analex-Denver could have used the real load tape values, but did not think they could.

In addition, Analex-Denver, in designing the IV&V process, did not understand the generation or internal verification process for all the constants in the "truth baseline" provided to them by LMA. The Analex-Denver engineers were not aware that the I1 filter rate values provided originated from a manual input and might not be the same as those subjected to independent V&V by Analex-Cleveland.

None of the participants was aware that nobody was testing the software with the actual load tape values nor that the default values they used did not match the real values.

Coordination: This was a classic case of coordination problems. Responsibility was diffused among the various partners, without complete coverage. In the end, nobody tested the load tape

[6] Note that almost identical words were used in the Ariane 501 accident report [71].

and everyone thought someone else was doing it.

8.3.10 Systems Engineering

System engineering at LMA was responsible for the identification and allocation of the functionality to be included in the system. In fact, the software filter involved in the loss was not needed and should have been left out instead of being retained, yet another example of asynchronous evolution. Why was that decision made? The filter was designed to prevent the Centaur from responding to the effects of Milstar fuel sloshing and inducing roll rate errors at 4 radians/second. Early in the design phase of the first Milstar satellite, the manufacturer asked to filter that frequency. The satellite manufacturer subsequently determined filtering was not required at that frequency and informed LMA. However, LMA decided to leave the filter in place for the first and subsequent Milstar flights for "consistency".[7] No further explanation is included in the report.

8.3.11 LMA Project Management (as Prime Contractor)

Process Being Controlled: The activities involved in the development and assurance of the system and its components.

Safety Constraint: Effective software development processes must be established and monitored. System safety processes must be created to identify and manage system hazards.

Context: The Centaur software process was developed early in the Titan/Centaur program: Many of the individuals who designed the original process were no longer involved in it due to corporate mergers and restructuring (e.g., Lockheed, Martin Marietta, General Dynamics) and the maturation and completion of the Titan IV design and development. Much of the system and process history and design rationale was lost with their departure.

Control Algorithm Flaws:

- A flawed software development process was designed. For example, no process was provided for creating and validating the flight constants.
- LMA, as prime contractor, did not exert adequate control over the development process. The Accident Investigation Board could not identify a single process owner responsible for understanding, designing, documenting, or controlling configuration and ensuring proper execution of the process.
- An effective system safety program was not created.
- An inadequate IV&V program (designed by Analex-Denver) was approved and instituted that did not verify or validate the I1 filter rate constants used in flight.

Mental Model Flaws: Nobody seemed to understand the overall software development process and apparently all had a misunderstanding about the coverage of the testing process.

[7] This factor is similar to the Ariane 501 loss, where unnecessary software code was left in for "commonality reasons" [71]

8.3.12 Defense Contract Management Command (DCMC)

Process Being Controlled: The report is vague, but apparently DCMC was responsible for contract administration, software surveillance, and overseeing the development process.

Control Inadequacies: The report says that DCMC approved an IV&V process with incomplete coverage and that there was a software quality assurance function operating at DCMC, but it operated without a detailed understanding of the overall process or program and therefore was ineffective.

Coordination: No information was provided in the accident report although coordination problems between SMC and DCMA may have been involved. Were each assuming the other was monitoring the overall process? What role did Aerospace Corporation play? Were there gaps in the responsibilities assigned to each of the many groups providing oversight here? How did the overlapping responsibilities fit together? What kind of feedback did DCMC use to perform their process monitoring?

8.3.13 Air Force (Program Office): Space and Missile Systems Center Launch Directorate (SMC)

Process Being Controlled: Management of the Titan/Centaur/Milstar development and launch control structures. SMC was responsible for "insight" and administration of the LMA contract.

Safety Constraint: SMC must ensure that the prime contractor creates an effective development and safety assurance program.

Context: Like 3SLS, the Air Force Space and Missile System Center Launch Directorate was transitioning from a task oversight to a process insight role and had, at the same time, undergone personnel reductions.

Control Algorithm Flaws:

- The SMC Launch Programs Directorate essentially had no personnel assigned to monitor or provide insight into the generation and verification of the software development process. The Program Office did have support from Aerospace Corporation to monitor the software development and test process, but that support had been cut by over 50 percent since 1994. The Titan Program Office had no permanently assigned civil service or military personnel nor full-time support to work the Titan/Centaur software. They decided that because the Titan/Centaur software was "mature, stable, and had not experienced problems in the past" they could best use their resources to address hardware issues.

- The transition from oversight to insight was not managed by a detailed plan. AF responsibilities under the insight concept had not been well defined, and requirements to perform those responsibilities had not been communicated to the workforce. In addition, implementation of the transition from an oversight role to an insight role was negatively affected by the lack of documentation and understanding of the software development and testing process.

Similar flawed transitions to an "insight" role are a common factor in many recent aerospace accidents.

- The Titan Program Office did not impose any standards (e.g., Mil-Std-882) or process for safety. While one could argue about what particular safety standards and program could or should be imposed, it is clear from the complete lack of such a program that no guidance was provided. Effective control of safety requires that responsibility for safety be assigned at each level of the control structure. Eliminating this control leads to accidents. The report does not say whether responsibility for controlling safety was retained at the program office or whether it had been delegated to the prime contractor. But even if it had been delegated to LMA, the program office must provide overall leadership and monitoring of the effectiveness of the efforts. Clearly there was an inadequate safety program in this development and deployment project. Responsibility for detecting this omission lies with the program office.

In summary, understanding why this accident occurred and making the changes necessary to prevent future accidents requires more than simply identifying the proximate cause—a human error in transcribing long strings of digits. This type of error is well known and there should have been controls established throughout the process to detect and fix it. Either these controls were missing in the development and operations processes or they were inadequately designed and executed.

The next section presents an example of an accident analysis using STAMP where governmental and regulatory agencies played important roles in the accident process.

8.4 A Public Water Supply Contamination Accident

In May 2000, in the small town of Walkerton, Ontario, Canada, some contaminants, largely Escherichia coli O157:H7 (the common abbreviation for which is E. coli) and Campylobacter jejuni entered the Walkerton water system through a well of the Walkerton municipal water system. About half the people in the town of 4800 became ill and seven died [84]. First the proximate events are presented and then the STAMP analysis of the accident.

8.4.1 The Proximate Events at Walkerton

The Walkerton water system was operated by the Walkerton Public Utilities Commission (WPUC). Stan Koebel was the WPUC's general manager and his brother Frank its foreman. In May 2000, the water system was supplied by three groundwater sources: Wells 5, 6, and 7. The water pumped from each well was treated with chlorine before entering the distribution system.

The source of the contamination was manure that had been spread on a farm near Well 5. Unusually heavy rains from May 8 to May 12 carried the bacteria to the well. Between May 13 and May 15, Frank Koebel checked Well 5 but did not take measurements of chlorine residuals, although daily checks were supposed to be made.[8] Well 5 was turned off on May 15.

On the morning of May 15, Stan Koebel returned to work after having been away from Walkerton for more than a week. He turned on Well 7, but shortly after doing so, he learned a new chlorinator for Well 7 had not been installed and the well was therefore pumping unchlorinated water directly

[8] Low chlorine residuals are a sign that contamination is overwhelming the disinfectant capacity of the chlorination process.

into the distribution system. He did not turn off the well, but instead allowed it to operate without chlorination until noon on Friday May 19, when the new chlorinator was installed.

On May 15, samples from the Walkerton water distribution system were sent to A&L Labs for testing according to the normal procedure. On May 17, A&L Labs advised Stan Koebel that samples from May 15 tested positive for E. coli and total coliforms. The next day (May 18) the first symptoms of widespread illness appeared in the community. Public inquiries about the water prompted assurances by Stan Koebel that the water was safe. By May 19 the scope of the outbreak had grown, and a pediatrician contacted the local health unit with a suspicion that she was seeing patients with symptoms of E. coli.

The Bruce-Grey-Owen Sound (BGOS) Health Unit (the government unit responsible for public health in the area) began an investigation. In two separate calls placed to Stan Koebel, the health officials were told that the water was "okay." At that time, Stan Koebel did not disclose the lab results from May 15, but he did start to flush and superchlorinate the system to try to destroy any contaminants in the water. The chlorine residuals began to recover. Apparently, Mr. Koebel did not disclose the lab results for a combination of two reasons: he did not want to reveal the unsafe practices he had engaged in from May 15 to May 17 (i.e., running Well 7 without chlorination), and he did not understand the serious and potentially fatal consequences of the presence of E. coli in the water system. He continued to flush and superchlorinate the water through the following weekend, successfully increasing the chlorine residuals. Ironically, it was not the operation of Well 7 without a chlorinator that caused the contamination; the contamination instead entered the system through Well 5 from May 12 until it was shut down May 15.

On May 20, the first positive test for E. coli infection was reported and the BGOS Health Unit called Stan Koebel twice to determine whether the infection might be linked to the water system. Both times, Stan Koebel reported acceptable chlorine residuals and failed to disclose the adverse test results. The Health Unit assured the public that the water was safe based on the assurances of Mr. Koebel.

That same day, a WPUC employee placed an anonymous call to the Ministry of the Environment (MOE) Spills Action Center, which acts as an emergency call center, reporting the adverse test results from May 15. On contacting Mr. Koebel, the MOE was given an evasive answer and Mr. Koebel still did not reveal that contaminated samples had been found in the water distribution system. The Local Medical Officer was contacted by the health unit, and he took over the investigation. The health unit took their own water samples and delivered them to the Ministry of Health laboratory in London (Ontario) for microbiological testing.

When asked by the MOE for documentation, Stan Koebel finally produced the adverse test results from A&L Laboratory and the daily operating sheets for Wells 5 and 6, but said he could not produce the sheet for Well 7 until the next day. Later, he instructed his brother Frank to revise the Well 7 sheet with the intention of concealing the fact that Well 7 had operated without a chlorinator. On Tuesday May 23, Stan Koebel provided the altered daily operating sheet to the MOE. That same day, the health unit learned that two of the water samples it had collected on May 21 had tested positive for E. coli.

Without waiting for its own samples to be returned, the BGOS health unit on May 21 issued a boil water advisory on local radio. About half of Walkerton's residents became aware of the advisory on May 21, with some members of the public still drinking the Walkerton town water as late as May 23. The first person died on May 22, a second on May 23, and two more on May 24. During this time, many children became seriously ill and some victims will probably experience lasting damage to their kidneys as well as other long-term health effects. In all, seven people died and more than 2300 became ill.

Looking only at these proximate events and connecting them by some type of causal chain, it appears that this is a simple case of incompetence, negligence, and dishonesty by WPUC employees. In fact, the government representatives argued at the accident inquiry that Stan Koebel or the Walkerton Public Utilities Commission (PUC) were solely responsible for the outbreak and that they were the only ones who could have prevented it. In May 2003, exactly three years after the accident, Stan and Frank Koebel were arrested for their part in the loss. But a systems-theoretic analysis using STAMP provides a much more informative and useful understanding of the accident besides simply blaming it only on the actions of Koebel brothers.

8.4.2 The System Hazards, System Safety Constraints, and Control Structure

As in the previous examples, the first step in creating a STAMP analysis is to identify the system hazards, the system safety constraints, and the hierarchical control structure in place to enforce the constraints.

The system hazard related to the Walkerton accident is public exposure to E. coli or other health-related contaminants through drinking water. This hazard leads to the following system safety constraint:

The safety control structure must prevent exposure of the public to contaminated water.

1. *Water quality must not be compromised.*
2. *Public health measures must reduce risk of exposure if water quality is compromised (e.g., boil water advisories).*

Each component of the socio-technical public water system safety control structure plays a role in enforcing this general system safety constraint and will, in turn, have their own safety constraints to enforce that are related to the function of the particular component in the overall system. For example, the Canadian federal government is responsible for establishing a nationwide public health system and ensuring it is operating effectively. Federal guidelines are provided to the Provinces, but responsibility for water quality is primarily delegated to each individual Province.

The provincial governments are responsible for regulating and overseeing the safety of the drinking water. They do this by providing budgets to the ministries involved—in Ontario these are the Ministry of the Environment (MOE), the Ministry of Health (MOH), and the Ministry of Agriculture, Food, and Rural Affairs—and by passing laws and adopting government policies affecting water safety.

According to the report on the official Inquiry into the Walkerton accident [84], the Ministry of Agriculture, Food, and Rural Affairs in Ontario is responsible for regulating agricultural activities with potential impact on drinking water sources. In fact, there was no watershed protection plan to protect the water system from agricultural runoff. Instead, the Ministry of the Environment was responsible for ensuring that the water systems could not be affected by such runoff.

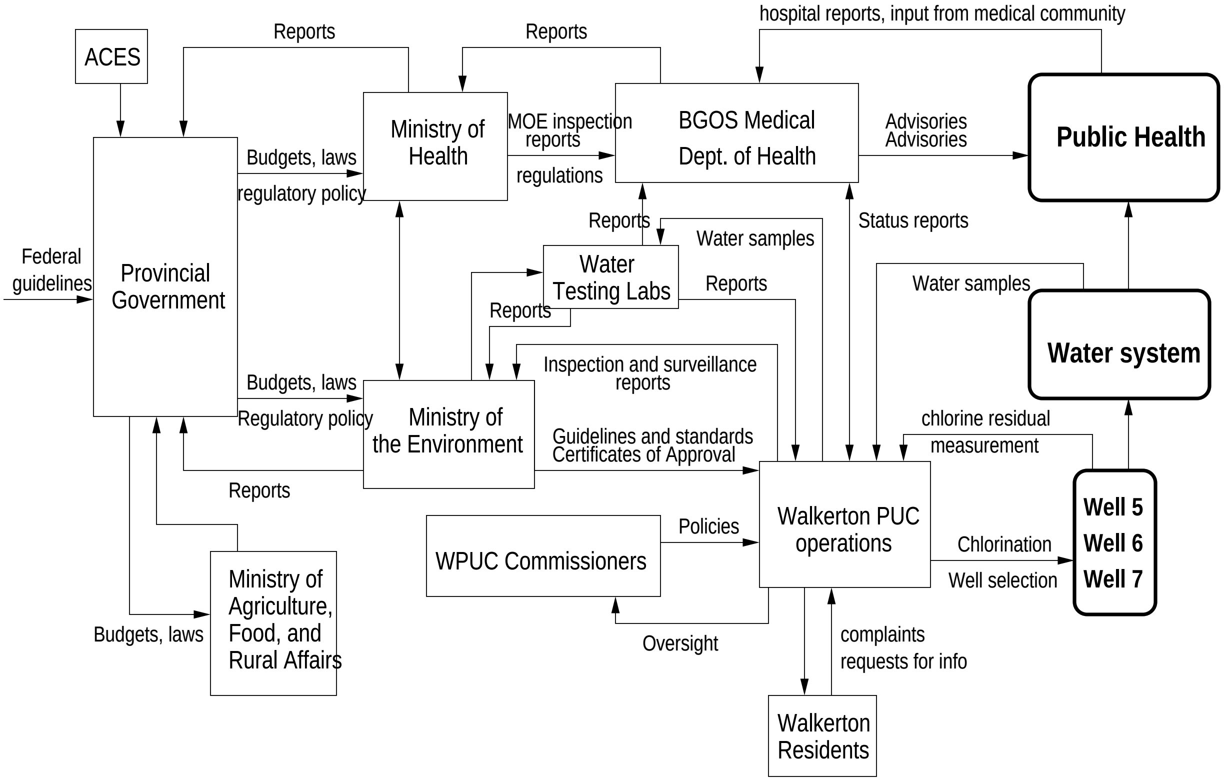
The Ministry of the Environment (MOE) has primary responsibility for regulating and for enforcing legislation, regulations, and policies that apply to the construction and operation of municipal water systems. Guidelines and objectives are set by the MOE, based on Federal guidelines. They are enforceable through Certificates of Approval issued to public water utilities operators, under the Ontario Water Resources Act. The MOE also has legislative responsibility for building

System Hazard: Public is exposed to e. coli or other health−related contaminants through drinking water.

System Safety Constraints: The safety control structure must prevent exposure of the public to contaminated water.

(1) Water quality must not be compromised.

(2) Public health measures must reduce risk of exposure if water quality is compromised (e.g., notification and procedures to follow)



Safety Requirements and Constraints:

Federal Government

- Establish a nationwide public health system and ensure it is operating effectively.

Provincial Government

- Establish regulatory bodies and codes of responsibilities, authority, and accountability
- Provide adequate resources to regulatory bodies to carry out their responsibilities.
- Provide oversight and feedback loops to ensure that provincial regulatory bodies are doing their job adequately.
- Ensure adequate risk assessment is conducted and effective risk management plans are in place.

Ministry of the Environment

- Ensure that those in charge of water supplies are competent to carry out their responsibilities.
- Perform inspections and surveillance. Enforce compliance if problems found.
- Perform hazard analyses to identify vulnerabilities and monitor them.
- Perform continual risk evaluation for existing facilities and establish new controls if necessary.
- Establish criteria for determining whether a well is at risk.
- Establish feedback channels for adverse test results. Provide multiple paths.
- Enforce legislation, regulations and policies applying to construction and operation of municipal water systems.
- Establish certification and training requirements for water system operators.

ACES

- Provide stakeholder and public review and input on ministry standards

Ministry of Health

- Ensure adequate procedures exist for notification and risk abatement if water quality is compromised.

Water Testing Labs

- Provide timely reports on testing results to MOE, PUC, and and Medical Dept. of Health

WPUC Commissioners

- Oversee operations to ensure water quality is not compromised.

WPUC Operations Management

- Monitor operations to ensure that sample taking and reporting is accurate and adequate chlorination is being performed.

WPUC Operations

- Measure chlorine residuals.
- Apply adequate doses of cholorine to kill bacteria.

BGOS Medical Department of Health

- Provide oversight of drinking water quality.
- Follow up on adverse drinking water quality reports.
- Issue boil water advisories when necessary.

Figure 8.10: The Basic Water Safety Control Structure. Lines going into the left of a box are control lines. Lines from or to the top or bottom of a box represent information, feedback, or a physical flow. Rectangles with sharp corners are controllers while rectangles with rounded corners represent plants.

and maintaining water treatment plants and has responsibility for public water system inspections and drinking water surveillance, for setting standards for certification of water systems, and for continuing education requirements for operators to maintain competence as knowledge about water safety increases.

The Ministry of Health supervises local Health Units, in this case, the Bruce-Grey-Owen-Sound (BGOS) Department of Health, run by local Officers of Health in executing their role in protecting public health. The BGOS Medical Dept. of Health receives inputs from various sources, including hospitals, the local medical community, the Ministry of Health, and the Walkerton Public Utilities Commission, and in turn is responsible for issuing advisories and alerts if required to protect public health. Upon receiving adverse water quality reports from the government testing labs or the MOE, the local public health inspector in Walkerton would normally contact the WPUC to ensure that followup samples were taken and chlorine residuals maintained.

The public water system in Walkerton is run by the Walkerton Public Utilities Commission (WPUC), which operates the wells and is responsible for chlorination and for measurement of chlorine residuals. Oversight of the WPUC is provided by elected WPUC Commissioners. The Commissioners are responsible for establishing and controlling the policies under which the PUC operates, while the general manager (Stan Koebel) and staff are responsible for administering these policies in operating the water facility. Although theoretically also responsible for the public water system, the municipality left the operation of the water system to the WPUC.

Together, the safety constraints enforced by all of these system control components must be adequate to enforce the overall system safety constraints. Figure 8.10 shows the overall theoretical water safety control structure in Ontario and the safety-related requirements and constraints for each system component.

Each component of the socio-technical public water safety system plays a role in enforcing the overall system safety constraint. Understanding the accident requires understanding the role in the accident scenario played by each level of the system's hierarchical control structure in the accident by not adequately enforcing its part of the safety constraint. The inadequate control (in terms of enforcing the safety constraints) exhibited by each component in the Walkerton accident is described in Sections 8.4.3, through 8.4.7. For each component, the contribution to the accident is described in terms of the four conditions required for adequate control, i.e., the goal, the actions, the process or mental models, and feedback. At each level of control, the context in which the behaviors took place is also considered. It is not possible to understand human behavior without knowing the context in which it occurs and the behavior-shaping factors in the environment.

This first level of analysis provides a view of the limitations of the static control structure at the time of the accident. But systems are not static—they adapt and change over time. In STAMP, systems are treated as a dynamic process that is continually adapting to achieve its ends and to react to changes in itself and its environment. The original system design must not only enforce the system safety constraints, but the system must continue to enforce the constraints as changes occur. The analysis of accidents, therefore, requires understanding not only the flaws in the static control structure that allowed the safety constraints to be violated but also the changes to the safety control structure over time (the *structural dynamics*) and the dynamic processes behind these changes (the *behavioral dynamics*). Section 8.4.8 analyzes the structural dynamics of the Walkerton accident while Section 8.4.9 shows the behavioral dynamics.

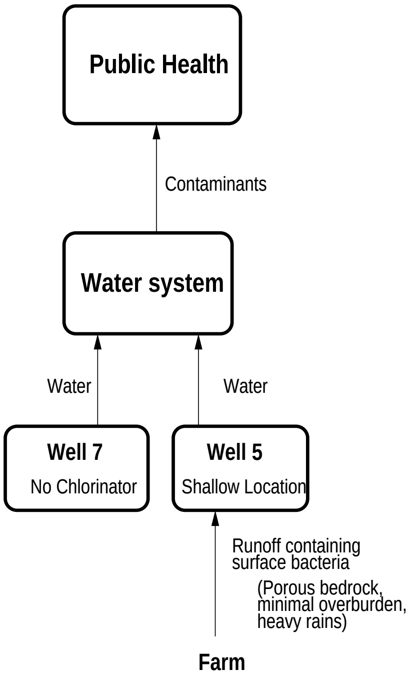


Figure 8.11: The Physical Components of the Water Safety Control Structure.

8.4.3 The Physical Process View of the Accident

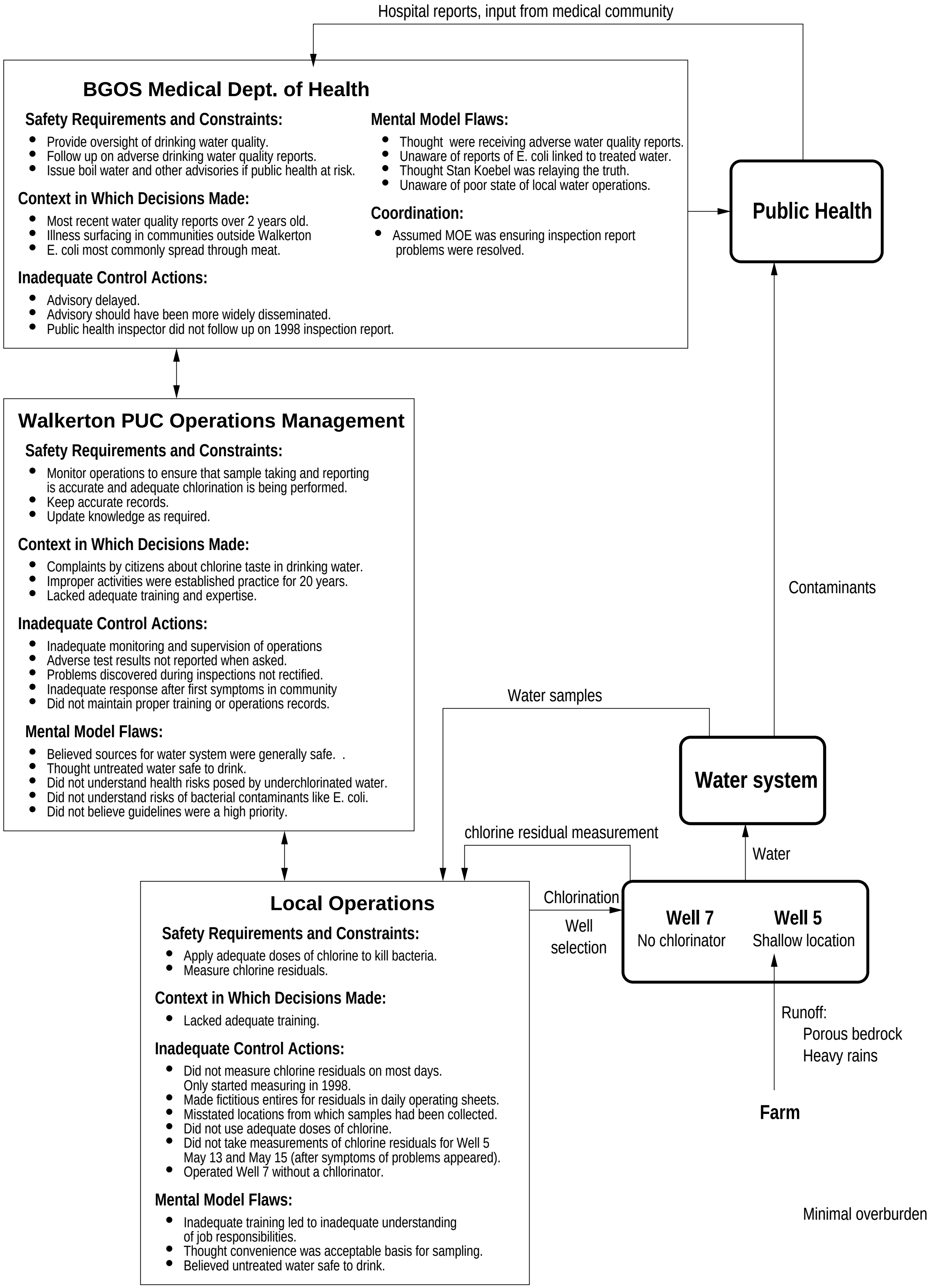
As in many system accidents, there were no physical failures involved. If, as in Figure 8.11, we draw the boundary of the physical system around the wells, the public water system, and public health, then one can describe the "cause" of the accident at the physical system level as the inability of the physical design to enforce the physical safety constraint in the face of an environmental disturbance, i.e., the unusually heavy rains that resulted in the transport of contaminants from the fields to the water supply. The safety constraint being enforced at this level is that water must be free from unacceptable levels of contaminants.

Well 5 was a very shallow well: all of its water was drawn from an area between 5m and 8m below the surface. More significantly, the water was drawn from an area of bedrock, and the shallowness of the soil overburden above the bedrock along with the fractured and porous nature of the bedrock itself made it possible for surface bacteria to make its way to Well 5.

8.4.4 The First Level Operations

Besides the physical system analysis, most hazard analysis techniques and accident investigations consider the immediate operators of the system. Figure 8.12 shows the results of a STAMP analysis of the flaws by the lower operations levels at Walkerton that were involved in the accident.

The safety requirements and constraints on the operators of the local water system were that they must apply adequate doses of chlorine to kill bacteria and must measure chlorine residuals. Stan Koebel, the WPUC Manager, and Frank Koebel, its foreman, were not qualified to hold their



BGOS Medical Dept. of Health

Safety Requirements and Constraints:

- Provide oversight of drinking water quality.
- Follow up on adverse drinking water quality reports.
- Issue boil water and other advisories if public health at risk.

Context in Which Decisions Made:

- Most recent water quality reports over 2 years old.
- Illness surfacing in communities outside Walkerton
- E. coli most commonly spread through meat.

Inadequate Control Actions:

- Advisory delayed.
- Advisory should have been more widely disseminated.
- Public health inspector did not follow up on 1998 inspection report.

Mental Model Flaws:

- Thought were receiving adverse water quality reports.
- Unaware of reports of E. coli linked to treated water.
- Thought Stan Koebel was relaying the truth.
- Unaware of poor state of local water operations.

Coordination:

- Assumed MOE was ensuring inspection report problems were resolved.

Walkerton PUC Operations Management

Safety Requirements and Constraints:

- Monitor operations to ensure that sample taking and reporting is accurate and adequate chlorination is being performed.
- Keep accurate records.
- Update knowledge as required.

Context in Which Decisions Made:

- Complaints by citizens about chlorine taste in drinking water.
- Improper activities were established practice for 20 years.
- Lacked adequate training and expertise.

Inadequate Control Actions:

- Inadequate monitoring and supervision of operations
- Adverse test results not reported when asked.
- Problems discovered during inspections not rectified.
- Inadequate response after first symptoms in community
- Did not maintain proper training or operations records.

Mental Model Flaws:

- Believed sources for water system were generally safe. .
- Thought untreated water safe to drink.
- Did not understand health risks posed by underchlorinated water.
- Did not understand risks of bacterial contaminants like E. coli.
- Did not believe guidelines were a high priority.

Local Operations

Safety Requirements and Constraints:

- Apply adequate doses of chlorine to kill bacteria.
- Measure chlorine residuals.

Context in Which Decisions Made:

- Lacked adequate training.

Inadequate Control Actions:

- Did not measure chlorine residuals on most days. Only started measuring in 1998.
- Made fictitious entires for residuals in daily operating sheets.
- Misstated locations from which samples had been collected.
- Did not use adequate doses of chlorine.
- Did not take measurements of chlorine residuals for Well 5 May 13 and May 15 (after symptoms of problems appeared).
- Operated Well 7 without a chllorinator.

Mental Model Flaws:

- Inadequate training led to inadequate understanding of job responsibilities.
- Thought convenience was acceptable basis for sampling.
- Believed untreated water safe to drink.

Figure 8.12: The Physical and Operational Components of the Water Safety Control Structure.

positions within the WPUC. Before 1993, there were no mandatory certification requirements and after 1993 they were certified through a grandfathering process based solely on experience. Mr. Koebel knew how to operate the water system mechanically, but he lacked knowledge about the health risks associated with a failure to properly operate the system and of the importance of following the requirements for treatment and monitoring of the water quality. The inquiry report stated that many improper operating practices had been going on for years before Stan Koebel became manager: He simply left them in place. These practices, some of which went back 20 years, included misstating the locations at which samples for microbial testing were taken, operating wells without chlorination, making false entries in daily operating sheets, not measuring chlorine residuals daily, not adequately chlorinating the water, and submitting false annual reports to the MOE.

The operators of the Walkerton water system did not intentionally put the public at risk. Stan Koebel and the other WPUC employees believed the untreated water was safe and often drank it themselves at the well sites. Local residents also pressed the WPUC to decrease the amount of chlorine used because they objected to the taste of chlorinated water.

A second first-level control component was the Local Health Units, in this case, the Bruce-Grey-Owen-Sound (BGOS) Department of Health. Local Health Units are supervised by the Ministry of Health and run by local Officers of Health to execute their role in protecting public health. The BGOS Medical Dept. of Health receives inputs (feedback) from various sources, including hospitals, the local medical community, the Ministry of Health, and the WPUC, and in turn is responsible for issuing advisories and alerts if required to protect public health. While the local Health Unit did issue a boil water advisory on local radio when they finally decided that the water system might be involved, this means of notifying the public was not very effective—other more effective means could have been employed. One reason for the delay was simply that evidence was not strong that the water system was the source of the contamination. E. coli is not often spread by meat, thus its common reference as the "hamburger disease." In addition, some reported cases of illness came from people who did not live in the Walkerton water district. Finally, the local health inspector had no reason to believe that there were problems with the way the Walkerton water system was operated.

An important event related to the accident occurred in 1996, when the government water testing laboratories were privatized. Previously, water samples were sent to government laboratories for testing. These labs then shared the results with the appropriate government agencies as well as the local operators. Upon receiving adverse water quality reports from the government testing labs or the MOE, the local public health inspector in Walkerton would contact the WPUC to ensure that followup samples were taken and chlorine residuals maintained.

After water testing laboratory services for municipalities were assumed by the private sector in 1996, the MOH Health Unit for the Walkerton area sought assurances from the MOE's local office that the Health Unit would continue to be notified of all adverse water quality results relating to community water systems. It received that assurance, both in correspondence and at a meeting, but it did not receive adverse water test reports and, therefore, without feedback about any problems in the water system, the local public health authorities assumed everything was fine.

In fact, there *were* warnings of problems. Between January and April of 2000 (the months just prior to the May E. Coli outbreak), the lab that tested Walkerton's water repeatedly detected coliform bacteria—an indication that surface water was getting into the water supply. The lab notified the MOE on five separate occasions. The MOE in turn phoned the WPUC, was assured the problems were being fixed, and let it go at that. The MOE did not inform the local Walkerton Medical Office of Health, however, as by law it was required to do.

The WPUC changed water testing laboratories in May 2000. The new laboratory, A&L Canada Laboratories East, was unaware of any notification guidelines. In fact, they considered test results to be confidential and thus improper to send to anyone but the client (in this case, the WPUC manager Stan Koebel).

In 1998, the BGOS Health Unit did receive a report on an MOE inspection of the Walkerton water system that showed some serious problems did exist. When the local Walkerton public health inspector read the report, he filed it, assuming that the MOE would ensure that the problems identified were properly addressed. Note the coordination problems here in an area of overlapping control. Both the MOE and the local public health inspector should have followed up on the 1998 inspection report, but there was no written protocol instructing the public health inspector on how to respond to adverse water quality or water system inspection reports. The MOE also lacked such protocols. Once again, the local public health authorities received no feedback that indicated water system operations were problematic.

Looking only at the physical system and local operations, it appears that the accident was simply the result of incompetent water system operators, who initially lied to protect their jobs (but who were unaware of the potentially fatal consequences of their lies) made worse by an inadequate response by the local Health Unit. If the goal is to find someone to blame, this conclusion is reasonable. If, however, the goal is to understand why the accident occurred in order to make effective changes (beyond simply firing the Koebel brothers) in order to prevent repetitions in the future or to learn how to prevent accidents in other situations, then a more complete study of the larger water safety control structure within which the local operations is embedded is necessary.

8.4.5 The Municipal Government

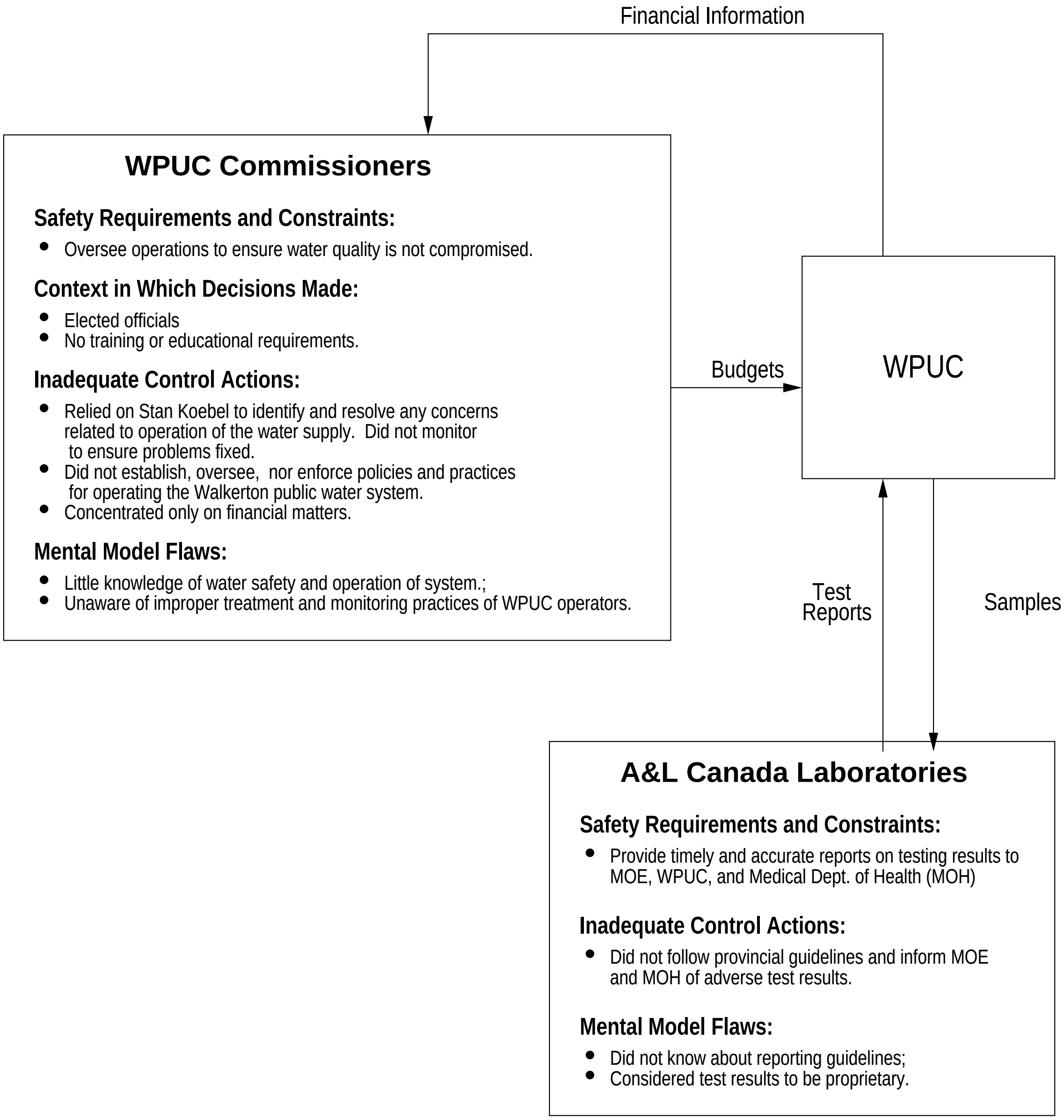
Figure 8.13 summarizes the flaws in the municipal water system control structure that allowed the dysfunctional interactions and thus the accident to occur.

Operating conditions on the public water system should theoretically have been imposed by the municipality, the Walkerton Public Utilities Commissioners, and the manager of the WPUC. The municipality left the operation of the water system to the WPUC. The WPUC Commissioners, who were elected, became over the years more focused on the finances of the PUC than the operations. They had little or no training or knowledge of water system operations or even water quality itself. Without such knowledge and with their focus on financial issues, they gave all responsibility for operations to the manager of the WPUC (Stan Koebel) and provided no other operational oversight.

The WPUC Commissioners received a copy of the 1998 inspection report but did nothing beyond asking for an explanation from Stan Koebel and accepting his word that he would correct the deficient practices. They never followed up to make sure he did. The mayor of Walkerton and the municipality also received the report but they assumed the WPUC would take care of the problems.

8.4.6 The Provincial Regulatory Agencies (Ministries)

The Ministry of the Environment (MOE) has primary responsibility for regulating and for enforcing legislation, regulations, and policies that apply to the construction and operation of municipal water systems. Guidelines and objectives are set by the MOE, based on Federal guidelines. They are



WPUC Commissioners

Safety Requirements and Constraints:

- Oversee operations to ensure water quality is not compromised.

Context in Which Decisions Made:

- Elected officials
- No training or educational requirements.

Inadequate Control Actions:

- Relied on Stan Koebel to identify and resolve any concerns related to operation of the water supply. Did not monitor to ensure problems fixed.
- Did not establish, oversee, nor enforce policies and practices for operating the Walkerton public water system.
- Concentrated only on financial matters.

Mental Model Flaws:

- Little knowledge of water safety and operation of system.;
- Unaware of improper treatment and monitoring practices of WPUC operators.

A&L Canada Laboratories

Safety Requirements and Constraints:

- Provide timely and accurate reports on testing results to MOE, WPUC, and Medical Dept. of Health (MOH)

Inadequate Control Actions:

- Did not follow provincial guidelines and inform MOE and MOH of adverse test results.

Mental Model Flaws:

- Did not know about reporting guidelines;
- Considered test results to be proprietary.

Figure 8.13: The Municipal Control Structure and Its Contribution to the Accident.

Ministry of the Environment

Safety Requirements and Constraints:

- Ensure those in charge of water supplies are competent to carry out their responsibilities.
- Perform inspections and enforce compliance if problems found.
- Perform hazard analyses to provide information about where vulnerabilities are and monitor them.
- Perform continual risk evaluation of existing facilities and establish new controls if necessary.
- Establish criteria for determining whether a well is at risk.
- Establish feedback channels for adverse test results. Provide multiple paths so that dysfunctional paths cannot prevent reporting.
- Enforce legislation, regulations, and policies applying to construction and operation of municipal water systems.
- Establish certification and training requirements for water system operators.

Context in Which Decisions Made:

- Critical information about history of known vulnerable water sources not easily accessible.
- Budget cuts and staff reductions

Inadequate Control Actions:

- No legally enforceable measures taken to ensure that concerns identified in inspections are addressed. Weak response to repeated violations uncovered in periodic inspections.
- Relied on voluntary compliance with regulations and guidelines.
- No systematic review of existing certificates of approval to determine if conditions should be added for continuous monitoring.
- Did not retroactively apply new approvals program to older facilities when procedures changed in 1992.
- Did not require continuous monitoring of existing facilities when ODWO amended in 1994.
- MOE inspectors not directed to assess existing wells during inspections.
- MOE inspectors not provided with criteria for determining whether a given well was at risk. Not directed to examine daily operating sheets.
- Inadequate inspections and improperly structured and administered inspection program..
- Approval of Well 5 without attaching operating conditions or special monitoring or inspection requirements.
- No followup on inspection reports noting serious deficiencies.
- Did not inform Walkerton Medical Officer of Health about adverse test results in January to April 2000 as required to do.
- Private labs not informed about reporting guidelines.
- No certification or training requirements for grandfathered operators.
- No enforcement of continuing training requirements.
- Inadequate training of MOE personnel.

Mental Model Flaws:

- Incorrect model of state of compliance with water quality regulations and guidelines.
- Several local MOE personnel did not know E. coli could be fatal.

Feedback:

- Did not monitor effects of privatization on reporting of adverse test results.
- Inadequate feedback about state of water quality and water test results.

Coordination:

- Neither MOE nor MOH took responsibility for enacting notification legislation.

Ministry of Health

Safety Requirements and Constraints:

- Ensure adequate procedures exist for notificatoin and risk abatement if water quality is compromised.

Inadequate Control Actions:

- No written protocol provided to local public health inspector on how to respond to adverse water quality or inspection reports.

Coordination:

- Neither MOE nor MOH took responsibility for enacting notification legislation.

Figure 8.14: The Role of the Ministries in the Accident.

enforceable through Certificates of Approval issued to public water utilities operators, under the Ontario Water Resources Act.

Walkerton Well 5 was built in 1978 and issued a Certificate of Approval by the MOE in 1979. Despite potential problems—the groundwater supplying the well was recognized as being vulnerable to surface contamination—no explicit operating conditions were imposed at the time.

Although the original Certificate of Approval for Well 5 did not include any special operating conditions, over time MOE practices changed. By 1992, the MOE had developed a set of model operating conditions for water treatment and monitoring that were routinely attached to new Certificates of Approval for municipal water systems. There was no effort, however, to determine whether such conditions should be attached to existing certificates, such as the one for Well 5.

The Provincial water quality guidelines were amended in 1994 to require the continuous monitoring of chlorine residuals and turbidity for wells supplied by a groundwater source that was under the direct influence of surface water (as was Walkerton's Well 5). Automatic monitoring and shutoff valves would have mitigated the operational problems at Walkerton and prevented the deaths and illness associated with the E. coli contamination in May 2000 if the requirement had been enforced in existing wells. However, at the time, there was no program or policy to review existing wells to determine whether they met the requirements for continuous monitoring. In addition, MOE inspectors were not directed to notify well operators (like the Koebel brothers) of the new requirement nor to assess during inspections if a well required continuous monitoring.

Stan and Frank Koebel lacked the training and expertise to identify the vulnerability of Well 5 themselves and to understand the resulting need for continuous chlorine residual and turbidity monitors. After the introduction of mandatory certification in 1993, the Koebel brothers were certified on the basis of experience even though they did not meet the certification requirements. The new rules also required 40 hours of training a year for each certified operator. Stan and Frank Koebel did not take the required amount of training, and the training they did take did not adequately address drinking water safety. The MOE did not enforce the training requirements and did not focus the training on drinking water safety.

The Koebel brothers and the Walkerton commissioners were not the only ones with inadequate training and knowledge of drinking water safety. Evidence at the Inquiry showed that several environmental officers in the MOE's local office were unaware that E. coli was potentially lethal and their mental models were also incorrect with respect to other matters essential to water safety.

At the time of the privatization of the government water testing laboratories in 1996, the MOE sent a guidance document to those municipalities that requested it. The document strongly recommended that a municipality include in any contract with a private lab a clause specifying that the laboratory directly notify the MOE and the local Medical Officer of Health about adverse test results. There is no evidence that the Walkerton PUC either requested or received this document. The MOE had no mechanism for informing private laboratories of the existing guidelines for reporting adverse results to the MOE and the MOH.

In 1997, the Minister of Health took the unusual step of writing to the Minister of the Environment requesting that legislation be amended to ensure that the proper authorities would be notified of adverse water test results. The Minister of the Environment declined to propose legislation, indicating that the existing guidelines dealt with the issue. On several occasions, officials in the MOH and the MOE expressed concerns about failures to report adverse test results to local Medical Officers of Health in accordance with the protocol. But the anti-regulatory culture and the existence of the Red Tape Commission discouraged any proposals to make notification legally binding on the operators or municipal water systems and private labs.

Another important impact of the 1996 law was a reduction in the MOE water system inspection program. The cutbacks at the MOE negatively impacted the number of inspections, although the inspection program had other deficiencies as well.

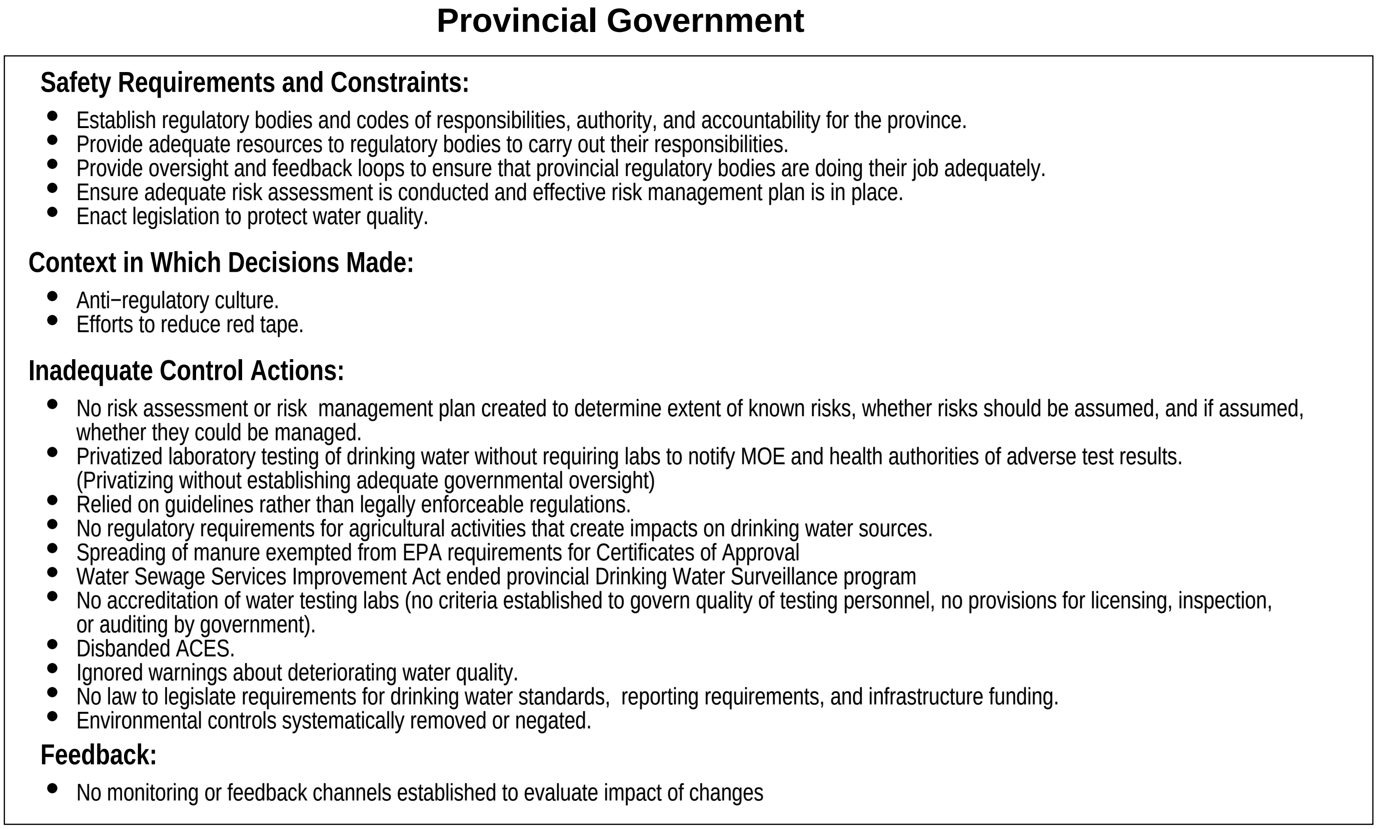
The MOE inspected the Walkerton water system in 1991, 1995, and 1998. At the time of the inspections, problems existed relating to water safety. Inspectors identified some of them, but unfortunately two of the most significant problems—the vulnerability of Well 5 to surface contamination and the improper chlorination and monitoring practices of the PUC—were not detected. Information about the vulnerability of Well 5 was available in MOE files, but inspectors were not directed to look at relevant information about the security of water sources and the archived information was not easy to find. Information about the second problem, improper chlorination and monitoring practices of the WPUC, was there to be seen in the operating records maintained by the WPUC. The Walkerton Inquiry report concludes that a proper examination of the daily operating sheets would have disclosed the problem. However, the inspectors were not instructed to carry out a thorough review of operating records.

The 1998 inspection report did show there had been problems with the water supply for years: detection of E. coli in treated water with increasing frequency, chlorine residuals in treated water at less than the required 0.5 mg/L, non-compliance with minimum bacteriological sampling requirements, and not maintaining proper training records.

The MOE outlined improvements that should be made, but desperately short of inspection staff and faced with small water systems across the province that were not meeting standards, it never scheduled a follow-up inspection to see if the improvements were in fact being carried out. The Walkerton Inquiry report suggests that the use of guidelines rather than regulations had an impact here. The report states that had the Walkerton PUC been found to be in non-compliance with a legally enforceable regulation, as opposed to a guideline, it is more likely that the MOE would have taken stronger measures to ensure compliance—such as the use of further inspections, the issuance of a Director's Order (which would have required the WPUC to comply with the requirements for treatment and monitoring), or enforcement proceedings. The lack of any followup or enforcement efforts may have led the Koebel brothers to believe the recommendations were not very important, even to the MOE.

Many adverse water quality reports were received from Walkerton between 1995 and 1998. During the mid to late 1990s, there were clear indications that the water quality was deteriorating. In 1996, for example, hundreds of people in Collingswood (a town near Walkerton), became ill after cryptosporidium (a parasite linked to animal feces) contaminated the drinking water. Nobody died, but it should have acted as a warning that the water safety control structure had degraded. Between January and April of 2000 (the months just prior to the May E. coli outbreak), the lab that tested Walkerton's water repeatedly detected coliform bacteria—an indication that surface water was getting into the water supply. The lab notified the MOE on five separate occasions. The MOE in turn phoned the WPUC, was assured the problems were being fixed, and let it go at that. The MOE failed to inform the Medical Officer of Health, as by law it was required to do.

Looking at the role of this hierarchical level in the Ontario water quality control system provides greater understanding of the reasons for the Walkerton accident and suggests more corrective actions that might be taken to prevent future accidents. But examining the control flaws at this level is not enough to understand completely the actions or lack of actions of the MOE. A larger view of the Provincial government role in the tragedy is necessary.



Provincial Government

Safety Requirements and Constraints:

- Establish regulatory bodies and codes of responsibilities, authority, and accountability for the province.
- Provide adequate resources to regulatory bodies to carry out their responsibilities.
- Provide oversight and feedback loops to ensure that provincial regulatory bodies are doing their job adequately.
- Ensure adequate risk assessment is conducted and effective risk management plan is in place.
- Enact legislation to protect water quality.

Context in Which Decisions Made:

- Anti−regulatory culture.
- Efforts to reduce red tape.

Inadequate Control Actions:

- No risk assessment or risk management plan created to determine extent of known risks, whether risks should be assumed, and if assumed, whether they could be managed.
- Privatized laboratory testing of drinking water without requiring labs to notify MOE and health authorities of adverse test results. (Privatizing without establishing adequate governmental oversight)
- Relied on guidelines rather than legally enforceable regulations.
- No regulatory requirements for agricultural activities that create impacts on drinking water sources.
- Spreading of manure exempted from EPA requirements for Certificates of Approval
- Water Sewage Services Improvement Act ended provincial Drinking Water Surveillance program
- No accreditation of water testing labs (no criteria established to govern quality of testing personnel, no provisions for licensing, inspection, or auditing by government).
- Disbanded ACES.
- Ignored warnings about deteriorating water quality.
- No law to legislate requirements for drinking water standards, reporting requirements, and infrastructure funding.
- Environmental controls systematically removed or negated.

Feedback:

- No monitoring or feedback channels established to evaluate impact of changes

Figure 8.15: The Role of the Provincial Government in the Accident.

8.4.7 The Provincial Government

The last component in the Ontario water quality control structure is the Provincial government. Figure 8.15 summarizes its role in the accident.

All of the weaknesses in the water system operations at Walkerton (and other municipalities) might have been mitigated if the source of contamination of the water had been controlled. A weakness in the basic Ontario water control structure was the lack of a government watershed and land use policy for agricultural activities that can impact drinking water sources. In fact, at a meeting of the Walkerton town council in November 1978 (when Well 5 was constructed), MOE representatives suggested land use controls for the area around Well 5, but the municipality did not have the legal means to enforce such land use regulations because the government of Ontario had not provided the legal basis for such controls.

At the same time as the increase in factory farms was overwhelming the ability of the natural filtration process to prevent the contamination of the local water systems, the spreading of manure had been granted a long-standing exemption from EPA requirements. Annual reports of the Environment Commissioner of Ontario for the four years before the Walkerton accident included recommendations that the government create a groundwater strategy. A Health Canada study stated that the cattle counties of Southwestern Ontario, where Walkerton is located, are high-risk areas for E. coli infections. The report pointed out the direct link between cattle density and E. coli infection, and showed that 32 percent of the wells in rural Ontario showed fecal contamination. Dr. Murray McQuigge, the Medical Officer of Health for the BGOS Health Unit (and the man who handled the Walkerton E. coli outbreak) warned in a memo to local authorities that "poor nutrient management on farms is leading to a degradation of the quality of ground water, streams, and lakes." Nothing was done in response.

With the election of a conservative provincial government in 1995, a bias against environmental regulation and red tape led to the elimination of many of the government controls over drinking water quality. A Red Tape Commission was established by the provincial government to minimize reporting and other requirements on government and private industry. At the same time, the government disbanded groups like the Advisory Committee on Environmental Standards (ACES), which reviewed ministry standards, including those related to water quality. At the time of the Walkerton contamination, there was no opportunity for stakeholder or public review of the Ontario clean water controls.

Budget and staff reductions by the conservative government took a major toll on environmental programs and agencies (although budget reductions had started before the election of the new provincial government). The MOE budget was reduced by 42% and 900 of the 2400 staff responsible for monitoring, testing, inspection, and enforcement of environmental regulations were laid off. The official Walkerton Inquiry report concludes that the reductions were not based on an assessment of the requirements to carry out the MOE's statutory requirements nor on any risk assessment of the potential impact on the environment or, in particular, on water quality. After the reductions, the Provincial Ombudsman issued a report saying that cutbacks had been so damaging that the government was no longer capable of providing the services that it was mandated to provide. The report was ignored.

In 1996, the Water Sewage Services Improvement Act was passed, which shut down the government water testing laboratories, downloaded control of provincially owned water and sewage plants to the municipalities, eliminated funding for municipal water utilities, and ended the provincial Drinking Water Surveillance Program, under which the MOE had monitored drinking water across the province.

The Provincial water quality guidelines directed testing labs to report any indications of unsafe water quality to the MOE and to the local Medical Officer Of Health. The latter would then decide whether to issue a boil water advisory. When government labs conducted all of the routine drinking water tests for municipal water systems throughout the province, it was acceptable to keep the notification protocol in the form of a guideline rather than a legally enforceable law or regulation. However, the privatization of water testing and the exit of government labs from this duty in 1996 made the use of guidelines ineffective in ensuring necessary reporting would occur. At the time, private environmental labs were not regulated by the government. No criteria were established to govern the quality of testing or the qualifications or experience of private lab personnel, and no provisions were made for licensing, inspection, or auditing of private labs by the government. In addition, the government did not implement any program to monitor the effect of privatization on the notification procedures followed whenever adverse test results were found.

In 1997, the Minister of Health took the unusual step of writing to the Minister of the Environment requesting that legislation be amended to ensure that the proper authorities would be notified of adverse water test results. The Minister of the Environment declined to propose legislation, indicating that the Provincial water quality guidelines dealt with the issue. On several occasions, officials in the MOH and the MOE expressed concerns about failures to report adverse test results to local Medical Officers of Health in accordance with the protocol. But the anti-regulatory culture and the existence of the Red Tape Commission discouraged any proposals to make notification legally binding on the operators or municipal water systems and private labs.

A final important change in the safety control structure involved the drinking water surveillance program in which the MOE monitored drinking water across the province. In 1996, the Provincial government dropped E. coli testing from its Drinking Water Surveillance Program. The next year, the Drinking Water Surveillance Program was shut down entirely. At the same time, the provincial government directed MOE staff not to enforce dozens of environmental laws and regulations still on the books. Farm operators, in particular, were to be treated with understanding if they were discovered to be in violation of livestock and waste-water regulations. By June, 1998, the Walkerton town council was concerned enough about the situation to send a letter directly to the Premier (Mike Harris), appealing for the province to resume testing of municipal water. There was no reply.

MOE officials warned the government that closing the water testing program would endanger public health. Their concerns were dismissed. In 1997, senior MOE officials drafted another memo that the government *did* heed [26]. This memo warned that cutbacks had impaired the Ministry's ability to enforce environmental regulations to the point that the Ministry could be exposed to lawsuits for negligence if and when an environmental accident occurred. In response, the Provincial government called a meeting of the Ministry staff to discuss how to protect itself from liability, and it passed a Bill ("The Environmental Approvals Improvement Act") which, among other things, prohibited legal action against the government by anyone adversely affected by the Environment Minister's failure to apply environmental regulations and guidelines.

Many other groups warned senior government officials, ministers, and the Cabinet of the danger of what it was doing, such as reducing inspections and not making the notification guidelines into regulations. The warnings were ignored. Environmental groups prepared briefs. The Provincial Auditor, in his annual reports, criticized the MOE for deficient monitoring of groundwater resources and for failing to audit small water plants across the province. The International Joint Commission expressed its concerns about Ontario's neglect of water quality issues, and the Environmental Commissioner of Ontario warned that the government was compromising environmental protection, pointing specifically to the testing of drinking water as an area of concern.

In January 2000 (three months before the Walkerton accident), staff at the MOE's Water Policy Branch submitted a report to the Provincial government warning that "Not monitoring drinking water quality is a serious concern for the Ministry in view of its mandate to protect public health." The report stated that a number of smaller municipalities were not up to the job of monitoring the quality of their drinking water. It further warned that because of the privatization of the testing labs, there was no longer a mechanism to ensure that the MOE and the local Medical Officer of Health were informed if problems were detected in local water systems. The Provincial government ignored the report.

The warnings were not limited to groups or individuals. Many adverse water quality reports had been received from Walkerton between 1995 and 1998. During the mid to late 1990s, there were clear indications that the water quality was deteriorating. In 1996, for example, hundreds of people in Collingswood (a town near Walkerton), became ill after cryptosporidium (a parasite linked to animal feces) contaminated the drinking water. Nobody died, but it should have acted as a warning that the water safety control structure had degraded.

The Walkerton Inquiry report notes that the decisions to remove the water safety controls in Ontario or to reduce their enforcement were taken without an assessment of the risks or the preparation of a risk management plan. The report says there was evidence that those at the most senior levels of government who were responsible for the decisions considered the risks to be manageable, but there was no evidence that the specific risks were properly assessed or addressed.

Up to this point, the Walkerton accident has been viewed in terms of inadequate control and enforcement of safety constraints. But systems are not static. The next two sections describe the dynamic aspects of the accident.

8.4.8 Understanding the Structural Dynamics of Accidents

Most hazard analysis and other safety engineering techniques treat systems and their environments as a static design. But systems are never static: They are continually adapting and changing to achieve their ends and to react to changes within themselves, in their goals, and in their environment. The original design must not only enforce appropriate constraints on behavior to ensure safe operation, but it must continue to operate safely as changes and adaptations occur over time. Accidents in a systems-theoretic framework are viewed as the result of flawed processes and control structures that evolve over time.

This tendency for the safety control structure to change over time accounts for the observation in Part I of this book that accidents in complex systems frequently involve a migration of the system toward a state where a small deviation (in the physical system, human operator behavior, or the environment) can lead to a catastrophe. The foundation for an accident is often laid years before. One event may trigger the loss, but if that event had not happened, another one would have triggered it. The safety control structure may degrade over time without any particular single decision to increase risk but simply as a series of decisions that move the plant slowly toward a situation where any slight error will lead to a major accident.

Humans and organizations can adapt and still maintain a low level of risk as long as the adaptations do not involve degradation in the control structure enforcing the system safety constraints. If this degradation does occur, however, the system moves toward states of increasing risk until an accident is triggered. The key here is that adaptation is not a random process. Instead, it is an optimization process, and therefore should be predictable and potentially controllable. The accident model, in order to handle system adaptation over time, must consider the processes involved in accidents: Processes control a sequence of events and describe system and human behavior as it changes and adapts over time rather than considering individual events and human actions.

Asynchronous evolution is one type of dysfunctional adaptation (as described in Part II) that can lead to degradation of the safety-control structure [63]. In asynchronous evolution, one part of a system changes without the related necessary changes in other parts. Changes to subsystems may be carefully designed, but consideration of their effects on other parts of the system, including the control aspects, may be neglected or inadequate. Asynchronous evolution may also occur when one part of a properly designed system deteriorates. In both these cases, the erroneous expectations of users or system components about the behavior of the changed or degraded subsystem may lead to accidents.

The public water safety control structure in Ontario started out with some weaknesses, which were mitigated by the presence of other controls. In some cases, the control over hazards was improved over time, for example, by the introduction of operator certification requirements and by requirements added in 1994 for continuous monitoring or chlorine residuals and turbidity in wells directly influenced by surface water. While these improvements were helpful for new wells, the lack of a policy to apply them to the existing wells and existing operators left serious weaknesses in the overall public health structure.

At the same time, other actions, such as the reduction in inspections and the elimination of the surveillance program reduced the feedback to the MOE and the MOH about the state of the system components. The water testing laboratory privatization by itself did not degrade safety, it was the way the privatization was implemented, i.e., without mandatory requirements for the private testing labs to inform the government agencies about adverse test results and without informing the private labs about the guidelines for this notification. Without regulations or oversight or enforcement of safe operating conditions, and with inadequate mental models of the safety requirements, operating practices have a tendency to change over time in order to optimize a variety of goals that conflict with safety, in this case, cutting budgets, reducing government, and reducing red tape.

An example of asynchronous evolution of the control structure is the assumption by the municipal government (Mayor and City Council) that appropriate oversight of the public water system operations was being done by the WPUC Commissioners. This assumption was true for the early operations. But the elected Commissioners over time became more interested in budgets and less expert in water system operation until they were not able to provide the necessary oversight. The municipal government, not understanding the changes, did not make an appropriate response.

Changes may also involve the environment. The lack of a Provincial watershed protection plan was compensated for by the Ministry of the Environment ensuring that the water systems could not be affected by such runoff. The original Walkerton design satisfied this safety constraint. But factory farms and farming operations increased dramatically and the production of animal waste overwhelmed the existing design safeguards. The environment had changed, but the existing controls were not revisited to determine whether they were still adequate. The system safety control structure had not changed in response to the changes in the environment, allowing an unusual but possible event (in this case, unusually heavy rain) to lead to a tragedy.

All of these changes in the Ontario water safety control structure over time led to the modified control structure shown in Figure 8.16. Dotted lines represent communication, control or feedback channels that still existed but had become ineffective. One thing to notice in comparing the original structure at the top and the one at the bottom is the disappearance of many of the feedback loops.

8.4.9 Modeling the Behavioral Dynamics of the Walkerton Accident[9]

As discussed in the previous section, the system's defenses or safety controls may degrade over time due to changes in the behavior of the components of the safety control loop. The reasons for the migration of the system toward a state of higher risk will be system specific and can be quite complex. In contrast to the usually simple and direct relationships represented in event chain accident models, most accidents in complex systems involve relationships between events and human actions that are highly non-linear, involving multiple feedback loops. The analysis or prevention of these accidents requires an understanding not only of the static structure of the system and of the

[9] Parts of this section were written by Mirna Daouk, Nicolas Dulac, and Karen Marais, who created the systems dynamics model of the Walkerton accident. We are working on making these models easier to read and to create so they can be used more widely by engineers without extensive training but the results of this research are not yet ready for this book and will be added later.

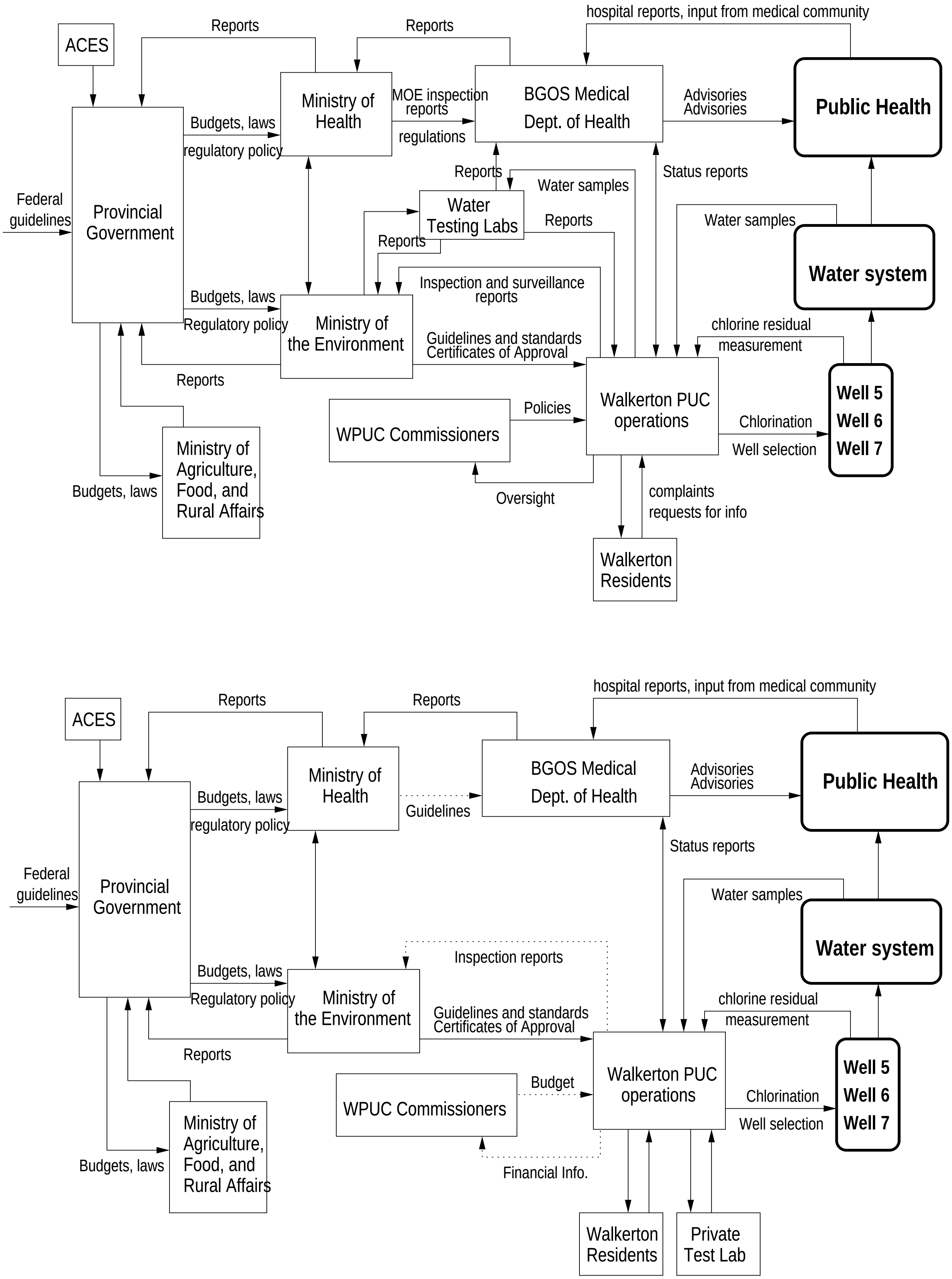


Figure 8.16: The Theoretical water safety control structure (Top) and the structure existing at the time of the accident (bottom). Note the elimination of many feedback loops.

changes to this structure over time (the *structural dynamics*), but also the dynamics behind these changes (the *behavioral dynamics*).

The previous section presented an approach to describing and analyzing the static safety control structure and how to use that to describe the changes to that structure that occur over time. This section presents a way to model and understand the dynamic processes *behind* the changes to the static control structure and *why* it changed over time, potentially leading to ineffective controls and unsafe or hazardous states.

The approach proposed uses the modeling techniques of *system dynamics.* The field of system dynamics, created at MIT in the 1950's by Jay Forrester, is designed to help decision makers learn about the structure and dynamics of complex systems, to design high leverage policies for sustained improvement, and to catalyze successful implementation and change. Drawing on engineering control theory and the modern theory of nonlinear dynamical systems, system dynamics involves the development of formal models and simulators to capture complex dynamics and to create an environment for organizational learning and policy design.

These ideas are particularly relevant when analyzing system accidents. The world is dynamic, evolving, and interconnected, but we tend to make decisions using mental models that are static, narrow, and reductionist [110]. Thus decisions that might appear to have no effect on safety—or even appear to be beneficial—may in fact degrade safety and increase risk. Using system dynamics, one can, for example, understand and predict instances of policy resistance or the tendency for well-intentioned interventions to be defeated by the response of the system to the intervention itself.

Figure 8.17 shows a system dynamics model for the Walkerton accident. The basic structures in the model are variables, stocks (represented by rectangles), and flows (double arrows into and out of stocks). Lines with arrows between the structures represent causality links, with a positive polarity meaning that a change in the original variable leads to a change in the same direction in the target variable. Similarly, a negative polarity means that a change in the original variable leads to a change in the opposite direction of the target variable. Double lines across a link represent a delay. Delays introduce the potential for instabilities in the system.

Modeling the entire systems' dynamics is usually impractical. The challenge is to choose relevant subsystems and model them appropriately for the intended purpose. STAMP provides the guidance for determining what to model when the goal is risk management. In the example provided, we focused primarily on the organizational factors, excluding the physical processes allowing the mixing of manure with the source water. Depending on the scope or purpose of the model, different processes could be added or removed.

According to systems dynamics theory, all the behavior dynamics of the system, despite their complexity, arise from two types of feedback loops [110]: positive (reinforcing) and negative (balancing). In system dynamics terms, degradation over time of the safety control structure, as represented by reinforcing loops, would lead inevitably to an accident, but there are balancing loops, such as regulation and oversight, that control those changes. In Ontario, as feedback and monitoring controls were reduced, the mental model of the central government leaders and the ministries responsible for water quality about the current state of the water system became increasingly divorced from reality. A belief that the water quality controls were in better shape than they actually were led to disregarding warnings and continued reduction in what were regarded as unnecessary regulation and red tape.

Accidents occur when the balancing loops do not adequately overcome the influences degrading the safety controls. Understanding why this degradation occurred (why risk increased) is an im-

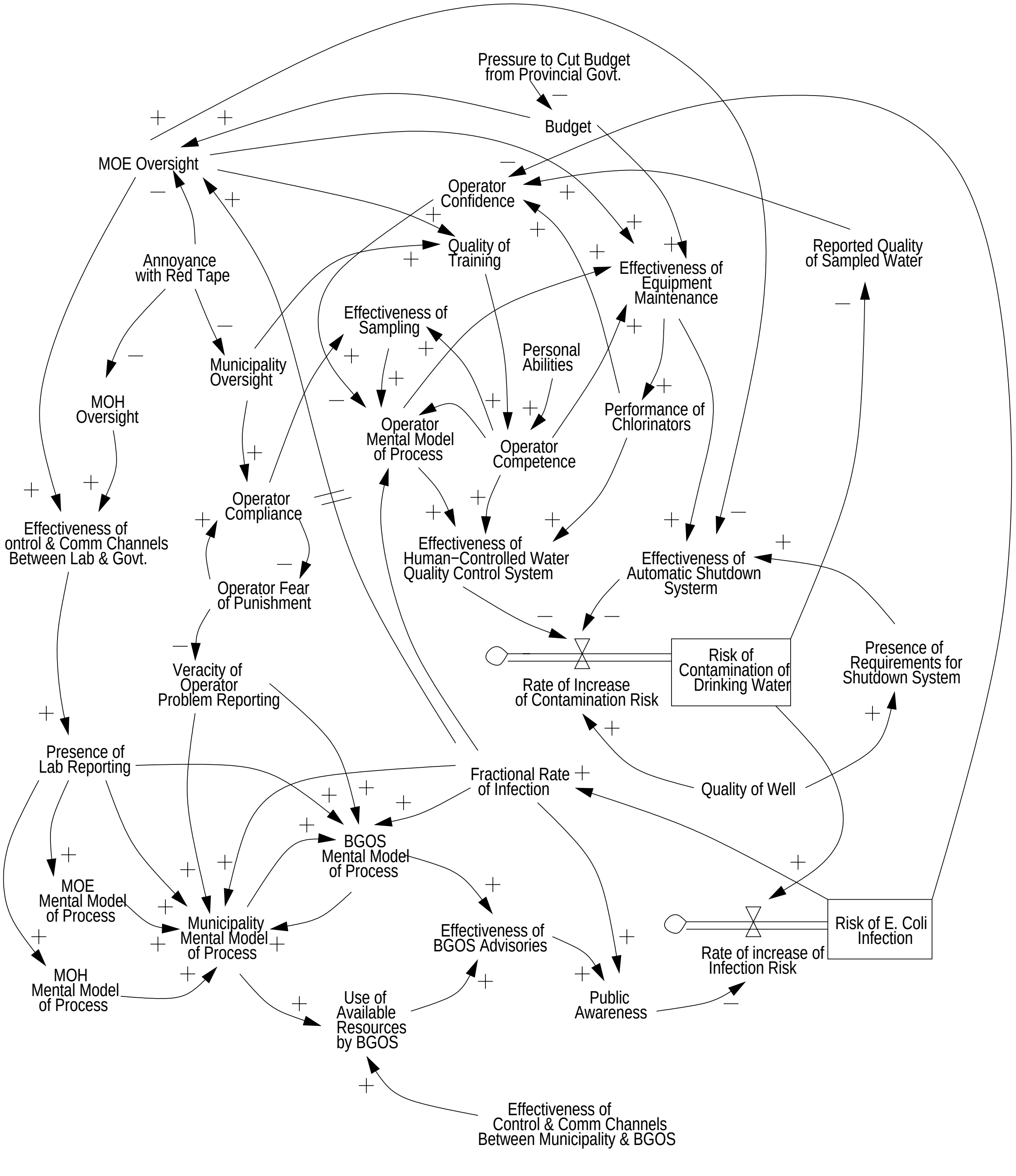


Figure 8.17: A Systems Dynamics Model for the Walkerton Water Contamination Accident

portant part of understanding why the accident occurred and learning how to prevent repetitions in the future, i.e. how to set up more effective safety control structures. It is also an important part of identifying when the socio-technical system is moving toward a state of unacceptable risk.

Our Walkerton model includes a number of exogenous variables (pressure to cut budgets, attempts by a conservative government to reduce business and government red tape, etc.) that act as levers on the behaviors of the system. When these variables are changed without any consideration of the dynamics of the system, the effectiveness of the safety control structure can deteriorate progressively, with few if any visible signs. For instance, the attempts to reduce red tape decreased the oversight of the ministries and municipalities. This decrease in oversight in turn had a negative effect on the control and communication channels between the government and the laboratories performing water quality analyses. Eventually, the laboratories stopped reporting the results of the tests. Because of this lack of reporting, the Walkerton municipality was much slower to realize that the water was contaminated, leading to a delay in the mobilization of the resources needed to deal with the contamination, and the effectiveness of the advisories issued was thus greatly diminished, increasing the risk of infection in the population.

Accident investigations often end with blame being assigned to particular individuals, often influenced by legal or political factors. The system dynamics models, on the other hand, can show how the attitude and behavior of individuals is greatly affected by the system structure and how and why such behavior may change over time. For instance, operator competence depends on the quality of training, which increases with government oversight but may decrease over time without such oversight due to competing pressures. An operator's fear of punishment, which in this case led Stan Koebel to lie about the adverse water quality test reports, is balanced by compliance with existing rules and regulations. This compliance, in turn, is directly influenced by the extent of government oversight and by the government's response to similar behavior in the past.

8.4.10 Addendum to the Walkerton Accident Analysis

The components included in the accident or safety analysis will obviously influence the causal factors considered. In the Walkerton accident, considering only the technical process and the immediate operators of the system, which are often the focus of accident investigation and prevention, provides only a limited view of the causal factors. If assigning blame in an accident investigation is the goal, then examining these two levels is usually enough to find someone that can be assigned responsibility for the accident. When designing safer systems is the goal rather than identifying who to punish, the emphasis needs to shift from *cause* (in terms of events or errors), which has a limiting, blame orientation, to understanding accidents in terms of *reasons*, i.e., why the events and errors occurred.

Each time we have presented this STAMP analysis of the Walkerton accident to a group, one or two members of the audience (usually employees of government regulatory agencies or accident investigation authorities) have objected and declared that the government actions were irrelevant and the accident cause was simply the actions of the Koebel brothers. Indeed, government representatives argued this point of view to the Walkerton accident investigators, although the author of the official Walkerton Inquiry report did not accept the viewpoint. Instead, the Inquiry report included recommendations to establish regulatory requirements for agricultural activities with potential impacts on drinking water sources, updating of standards and technology, improving current practices in setting standards, establishing legally enforceable regulations rather than guidelines, requiring mandatory training for all water system operators and requiring grandfathered operators to pass certification examinations within two years, developing a curriculum for operator training and mandatory training requirements specifically emphasizing water quality and safety issues, adopting a province-wide drinking water policy and a Safe Drinking Water Act, strictly enforcing drinking water regulations, and committing sufficient resources (financial and otherwise) to enable the MOE to play their role effectively. To date, most of these recommendations have not been implemented, but in May 2003 (exactly three years after the accident), the Koebel brothers were arrested for their part in the events. Water contamination incidents continue to occur in small towns in Ontario.

8.5 Conclusions

Accident models provide the basis for safety engineering—both in the analysis of accidents that have occurred and in the development of techniques to prevent accidents. This paper has suggested that dealing with safety in software-intensive systems will require more sophisticated accident models than those currently in use, which were developed for electromechanical systems. A proposal was made that models based on systems theory would be appropriate. One such model was described and illustrated using a software-related accident. Other types of models are possible.

A major difference between a systems-theoretic accident model and a chain-of-events model is that the former does not identify a root cause of an accident. Instead, the entire safety control structure is examined and the accident process to determine what role each part of the process played in the loss. While perhaps less satisfying in terms of assigning blame, the systems-theoretic analysis provides more information in terms of how to prevent future accidents.

Chapter 9

Identifying Hazards and Hazard Analysis

[*I have just started writing this chapter, and it is still very rough and incomplete. Missing sections will be covered in the class and class notes.*]

Chapter 8 described how to use STAMP in analyzing accidents and identifying causal factors. A major goal in any system program, however, is not just to investigate accidents after they occur and use that information to improve future safety but to prevent their occurrence in the first place.

In a classic system safety program, system hazards are first identified and then a careful analysis is performed to determine their potential causes. The resulting information is used in trade studies to evaluate the risk associated with various system designs and to eliminate or control hazards in the selected system design. The process of performing such hazard and risk analysis can be informally defined as investigating an accident before it occurs: STAMP provides a more powerful basis for performing such analysis in complex, software-intensive, and human-centered systems.

The first section of this chapter outlines the basic process of identifying system hazards and safety constraints. Then a new type of hazard analysis based on STAMP is described. The final section shows how risk can be considered in very early system concept definition and trade studies.

9.1 Identifying Hazards and Safety Constraints

The first step in any design for safety program should be the identification of the system hazards. To do this, accidents must be defined for the particular system being developed. An accident need not involve loss of life, but it does result in some loss that is unacceptable to the customers or users. For example, for spacecraft, accidents may include loss of the astronauts (if the spacecraft is manned), death or injury to the support personnel, nonaccomplishment of the mission, major equipment damage, environmental pollution, etc. Once an accident has been defined, the hazards associated with those losses can be identified.

To deal with complex systems, as emphasized continually throughout this book, accidents must be thought of as complex *processes* and prevention measures will involve the entire socio-technical system that can potentially have an affect on this process. Each socio-technical system component may control only part of the overall accident process. For example, an aircraft collision avoidance system like TCAS can only provide information and advisories that keep aircraft separated, but has no control over whether the pilots follow the advisories, the current weather and visibility conditions and the route of the aircraft with respect to them, performance of the aircraft, performance of the ground-based air traffic control system, etc. These other aspects are under the control of other components of the larger air traffic management system.

The hazards associated with a particular system component, therefore, are not equivalent to accidents but instead are factors involved in accidents. In a chemical plant, the hazard of releasing toxic chemicals into the atmosphere might not necessarily result in losses if the weather conditions lead to dissipation of the chemicals into the atmosphere. While the hazard could be defined as the death or injury of residents around the plant, there may be many factors involved in such a loss that are beyond the control of the plant designers and operators, such as atmospheric conditions (amount and direction of wind, etc.) at the time of the release.

For practical reasons, the hazards associated with any particular system component should be defined as conditions over which the system or subsystem designer has control (or at least partial control). If the larger socio-technical system is being considered, the number of hazards and potential actions to prevent them increases, for example, controlling land use near chemical plants or the siting of plants through zoning laws, and emergency evacuation and medical treatment planning. Each component of the socio-technical system may have different aspects of the accident under their control and thus be responsible for different parts of the accident process (hazards).

By considering the larger socio-technical system and not just the technical components, the most cost-effective way to eliminate or control hazards can be identified. If only part of the larger system is considered, the compromises required to eliminate or control the system hazard in that part of the overall system design may be much greater than would be necessary if other parts of the overall system are considered. For example, a particular hazard associated with launching a spacecraft might be controllable in the spacecraft design, in the physical launch infrastructure, by launch procedures, by the launch control system, or by a combination of these. If only the spacecraft design is considered in the hazard analysis process, hazard control may require more tradeoffs than if the hazard is partially or completely eliminated or controlled by design features in other parts of the launch process.

In addition, several components may have responsibilities related to the same hazards. The designers of the chemical plant and the relevant government agencies, for example, may both be concerned with plant design features potentially leading to inadvertent toxic chemical release. The government roles, however, may be restricted to design and construction approvals and inspection processes while the plant designers have basic design creation responsibilities. Consideration of hazards from the larger socio-technical will usually allow better solutions to be considered.

For practical reasons, a small set of high-level hazards should be identified first. Even complex systems usually have fewer than a dozen high-level hazards. Starting with too large a list at the beginning, usually caused by including refinements and causes of the high-level hazards in the list, leads to a disorganized and incomplete hazard identification and analysis process.

Consider aircraft collision control. The relevant accident is clearly a collision between two airborne aircraft. One (but only one) of the controls used to avoid this accident is airborne collision avoidance systems like TCAS II. The hazards related to TCAS II are:

1. TCAS causes or contributes to a near midair collision (NMAC), defined as a pair of controlled aircraft violating minimum separation standards.
2. TCAS causes or contributes to a controlled maneuver into the ground.
3. TCAS causes or contributes to the pilot losing control over the aircraft.

4. TCAS interferes with other safety-related aircraft systems.
5. TCAS interferes with the ground-based Air Traffic Control system (e.g., transponder transmissions to the ground or radar or radio services).
6. TCAS interferes with an ATC advisory that is safety-related (e.g., avoiding a restricted area or adverse weather conditions).

Ground-based Air Traffic Control also plays an important role in collision avoidance, although it has responsibility for a larger set of hazards:

1. Controlled aircraft violate minimum separation standards (NMAC).
2. An airborne controlled aircraft enters an unsafe atmospheric region.
3. A controlled airborne aircraft enters restricted airspace without authorization.
4. A controlled airborne aircraft gets too close to a fixed obstacle other than a safe point of touchdown on assigned runway (controlled flight into terrain or CFIT).
5. A controlled airborne aircraft and an intruder in controlled airspace violate minimum separation.
6. A controlled aircraft operates outside its performance envelope.
7. An aircraft on the ground comes too close to moving objects or collides with stationary objects or leaves the paved area.
8. An aircraft enters a runway for which it does not have a clearance.
9. A controlled aircraft executes an extreme maneuver within its performance envelope.
10. Loss of aircraft control.

A system is a recursive concept, that is, a system at one level may be viewed as a subsystem of a larger system. Unsafe behavior (hazards) at the system level can be translated into hazardous behaviors at the component or subsystem level. Note, however, that the reverse (bottom-up) process is not possible. Consider a train with an automated door system. The system level hazards related to the automated door system include a person being hit by closing doors, someone falling from a moving train or from a stationary train that is not properly aligned with a station platform, and passengers and staff being unable to escape from a dangerous environment in the train compartment (e.g., a fire). Translating these system hazards into the related hazardous behavior of the automated door system itself results in the following:

1. Door is open when the train starts.
2. Door opens while train is in motion.
3. Door opens while improperly aligned with station platform.
4. Door closes while someone is in the doorway.
5. Door that closes on an obstruction does not reopen or reopened door does not reclose.
6. Doors cannot be opened for emergency evacuation.

After the system and component hazards are identified, the next major goal is to identify the safety-related requirements and design constraints necessary to prevent the hazards from occurring. These constraints will be used in the system design and tradeoff analyses. When the system engineering process decomposes functions and allocates them to various system components, the constraints must also be refined and identified for each component. The process then iterates over the individual components.

Figure 9.1 shows an example of the design constraints that might be generated from the automated train door hazards. Note that the third constraint potentially conflicts with the last one

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

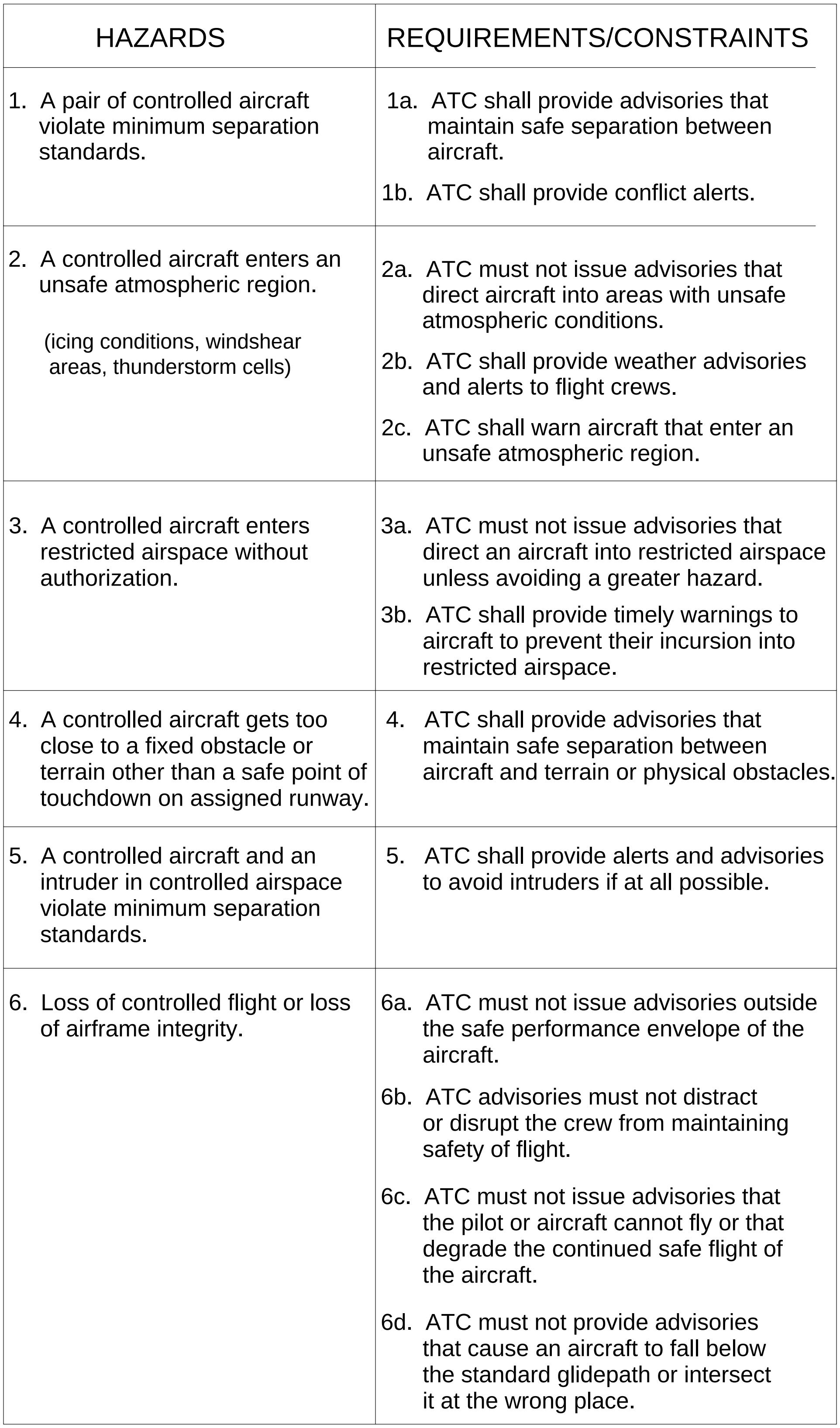
Figure 9.1: Design constraints for train door hazards

and the resolution of this conflict will be an important part of the system design process. Again, identifying these types of requirements and conflicts early in the design process will lead to better solutions. As the design process progresses and design decisions are made, these requirements and constraints will be refined and expanded. An example is shown in Chapter 11.

As a final example, consider the high-level safety-related design constraints for TCAS. These are shown in Figure [not yet created]. Figure 9.2 shows the high-level requirements and constraints for some of the air traffic control hazards identified above. Comparing the two is instructive. Some are closely related, such as the requirement to provide advisories that maintain safe separation between aircraft. This example of overlapping control raises important concerns about potential conflicts and coordination problems that need to be resolved. As noted in Chapter 7, accidents often occur in the boundary areas between controllers and when multiple controllers control the same process. The inadequate resolution of the conflict between multiple controller responsibilities for aircraft separation contributed to the collision of two aircraft over the town of Euberlingen (Germany) in July 2003 when TCAS and the ground air traffic controller provided conflicting advisories to the pilots.

Hazards can also be related to the interaction between components, for example the interaction between attempts by air traffic control to prevent collisions and the activities of pilots to maintain safe control over the aircraft (see constraint 6b in Figure 9.2). These types of potentially dysfunctional interactions obviously need to be considered during system design and included in the system hazard analysis.

How are the design constraints identified and refined? This process usually involves hazard analysis. The next section describes a new hazard analysis method, based on STAMP, to assist in accomplishing this goal.



HAZARDS	REQUIREMENTS/CONSTRAINTS
1. A pair of controlled aircraft violate minimum separation standards.	1a. ATC shall provide advisories that maintain safe separation between aircraft. 1b. ATC shall provide conflict alerts.
2. A controlled aircraft enters an unsafe atmospheric region. (icing conditions, windshear areas, thunderstorm cells)	2a. ATC must not issue advisories that direct aircraft into areas with unsafe atmospheric conditions. 2b. ATC shall provide weather advisories and alerts to flight crews. 2c. ATC shall warn aircraft that enter an unsafe atmospheric region.
3. A controlled aircraft enters restricted airspace without authorization.	3a. ATC must not issue advisories that direct an aircraft into restricted airspace unless avoiding a greater hazard. 3b. ATC shall provide timely warnings to aircraft to prevent their incursion into restricted airspace.
4. A controlled aircraft gets too close to a fixed obstacle or terrain other than a safe point of touchdown on assigned runway.	4. ATC shall provide advisories that maintain safe separation between aircraft and terrain or physical obstacles.
5. A controlled aircraft and an intruder in controlled airspace violate minimum separation standards.	5. ATC shall provide alerts and advisories to avoid intruders if at all possible.
6. Loss of controlled flight or loss of airframe integrity.	6a. ATC must not issue advisories outside the safe performance envelope of the aircraft. 6b. ATC advisories must not distract or disrupt the crew from maintaining safety of flight. 6c. ATC must not issue advisories that the pilot or aircraft cannot fly or that degrade the continued safe flight of the aircraft. 6d. ATC must not provide advisories that cause an aircraft to fall below the standard glidepath or intersect it at the wrong place.

Figure 9.2: High-level requirements and constraints for air traffic control

9.2 STAMP-Based Hazard Analysis (STPA)

Although STPA was first envisioned and developed for the technical parts of the system, we have since discovered that it works equally well on the social and organizational aspects. This section describes the use of STPA on technical components. Chapter 10 describes its use in an organizational risk analysis.

STPA starts in the early life cycle stages and continues through the life of the system. Its use during design can support a safety-driven design process where the hazard analysis influences and shapes the early system design decisions and then is iterated and refined as the design evolves and more information becomes available.

SFTA has the same general goals as any hazard analysis: (1) identification of the system hazards and the related safety constraints necessary to ensure acceptable risk and (2) accumulation of information about how those constraints could be violated so it can be used in eliminating, reducing, and controlling hazards.

The process starts with identifying the system requirements and design constraints necessary to maintain safety. In later steps, STPA assists in the top-down refinement of the safety-related system requirements and constraints into requirements and constraints on the individual system components. The overall process provides the information and documentation necessary to ensure the safety constraints are enforced in the system design, development, manufacturing, and operations.

The process of safety-driven design and the use of STPA to support it is illustrated here with the design of a non-existent Space Shuttle robotic Thermal Tile Processing System (TTPS)[1]. A specification of the example is contained in Appendix F. The TTPS will be responsible for inspecting and waterproofing the thermal protection tiles on the belly of the Shuttle.

The process starts, like any design process, with identifying general environmental constraints on the system design. These constraints derive from physical properties of the Orbital Processing Facility (OPF) at KSC, such as the size constraints on the physical system components and necessity of any mobile robotic components to deal with crowded work areas and for humans to be in the area. For example, the mobile robot must enter the facility through personnel access doors 1.1 m (42") wide. The layout within the OPF allows a length of 2.5 m (100") for the robot. There are some structural beams whose heights are as low as 1.75 m (70"), but once under the orbiter the tile heights range from about 2.9 meters to 4 meters. Thus the compact roll-in form of the mobile system must maneuver these spaces and also raise its inspection and injection equipment up to heights of 4 meters to reach individual tiles while meeting 1 mm accuracy requirements.

The next step, after the high-level system requirements (functional goals such as inspection and waterproofing) and environmental conditions have been identified, is to identify hazardous conditions that could be encountered during system operation (i.e., perform a preliminary hazard analysis).

Different initial system configurations could be chosen that would introduce different hazards. For our example, the initial configuration selected, given the identified high-level requirements and environmental constraints, consists of a robot containing a mobile base and a manipulator arm. As the concept and detailed design proceeds, information generated about hazards and design tradeoffs may lead to changes in the initial configuration. Alternatively, multiple design configurations may be considered in parallel.

[1] Nicolas Dulac created most of the example in this section

The next step is to identify the system hazards, such as contact of humans with waterproofing chemicals, fire or explosion, movement of the robot or manipulator arm causing injury to humans or damage to the orbiter, etc. Instability of the mobile base is used as the hazard considered in the rest of this section.

In general, safety-driven design involves first attempting to eliminate the hazard from the design and, if that is not possible or requires unacceptable tradeoffs, reducing the likelihood the hazard will occur, reducing the negative consequences of the hazard if it does occur, and implementing contingency plans for dealing with the hazard.

As design decisions are made, an STPA-based hazard analysis is used to inform those decisions. Early in the system design process, little information is available so the hazard analysis will be very general at first and will be refined and augmented as additional information emerges from the system design activities. For example, instability of the mobile base could lead to human injury or damage to the orbiter. A possible solution is to make the robot base so heavy that it cannot become unstable, no matter how the manipulator arm is positioned, thus eliminating the hazard. A heavy base, however, could increase the damage caused by the base coming into contact with a human or object or make it difficult for workers to manually move the robot out of the way in an emergency situation. An alternative solution is to make the base long and wide so the moment created by the operation of the manipulator is compensated by the moments created by base supports that are far from the robot's center of mass. A long and wide base could remove the hazard but may violate the environmental constraints in the facility layout (the need to maneuver through doors and in the crowded OPF).

Let's say that analysis of the environmental constraints results in a maximum length for the robot of 2.5 m and a width no larger than 1.1 m. Given the required maximum extension length of the manipulator arm and the weight of the equipment that will need to be carried, a simple analysis shows that the length of the robot base is sufficient to prevent any longitudinal instability, but the width of the base is clearly not sufficient to prevent lateral instability.

A solution that might be considered is the use of lateral stabilizer legs that are deployed when the manipulator arm is extended but must be retracted when the robot base moves.

At the initial stages, we identified only the general hazards, e.g., instability of the robot base and the related system design constraints that the mobile base must not be capable of falling over under worst-case operational conditions. As design decisions are proposed and analyzed, they will lead to additional refinements in the design constraints. For example, a solution to the stability problem is to use lateral stabilizer legs that are deployed when the manipulator arm is extended but must be retracted when the robot base moves. Under this scenario, two new safety design constraints are identified: (1) the manipulator arm must move only when the stabilizers are fully deployed and (2) the stabilizer legs must not be retracted until the manipulator arm is fully stowed. STPA is used to further refine these constraints and to evaluate the resulting designs.

STPA starts by defining an initial hierarchical control structure for the system. A candidate structure for the mobile robot is shown in figure 9.3. As with any part of the system design, the candidate control structure will have to be revisited when more information become available. In the candidate structure, a decision is made to introduce a human operator in order to supervise the robot during its operation and to perform safety-critical tasks. The STPA process will identify the implications of this decision and will assist in analyzing the allocation of tasks to the various components to determine the safety tradeoffs involved.

Using the initial control structure, the remaining activities in STPA are to identify poten-

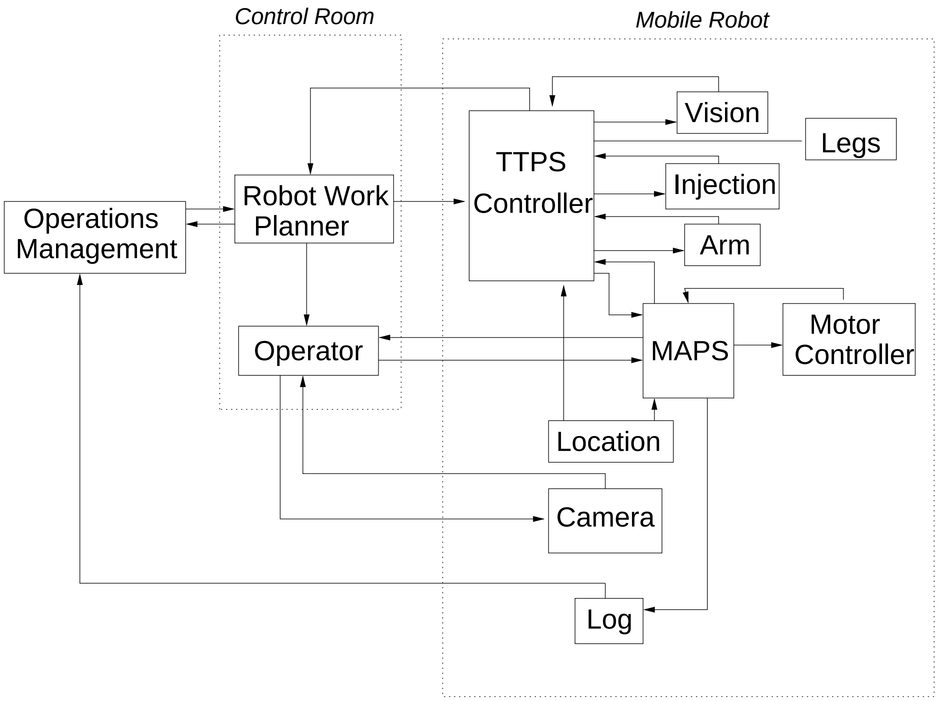


Figure 9.3: A Candidate Structure for the TTPS

tially hazardous control actions by each of the system components that could violate the safety constraints, determine the causal factors that could lead to these hazardous control actions, and prevent or control them in the system design. The process thus involves a top-down identification of scenarios in which the safety constraints could be violated so they can be used to guide design decisions. This general process is similar to fault tree analysis, but provides more types of scenarios and handles design errors better.

In general, a controller can provide four different types of inadequate control:

1. A required control action is not provided.
2. An incorrect or unsafe action is provided.
3. A potentially correct or adequate control action is provided too late or at the wrong time.
4. A correct control action is stopped too soon or continued too long.

For the TTPS mobile based and the preliminary design decisions described above, the stability constraint may be violated if the component responsible for controlling the position of the stabilizer legs:

1. Does not command a deployment of the stabilizer legs when arm movements are enabled.
2. Commands a retraction of the stabilizer legs when the manipulator arm is not stowed.
3. Commands a retraction of the stabilizer legs after arm movements are enabled or commands a retraction of the stabilizer legs before the manipulator arm is stowed.
4. Stops extension of the stabilizer legs before they are fully stowed.

These inadequate control actions can be restated as system safety constraints:

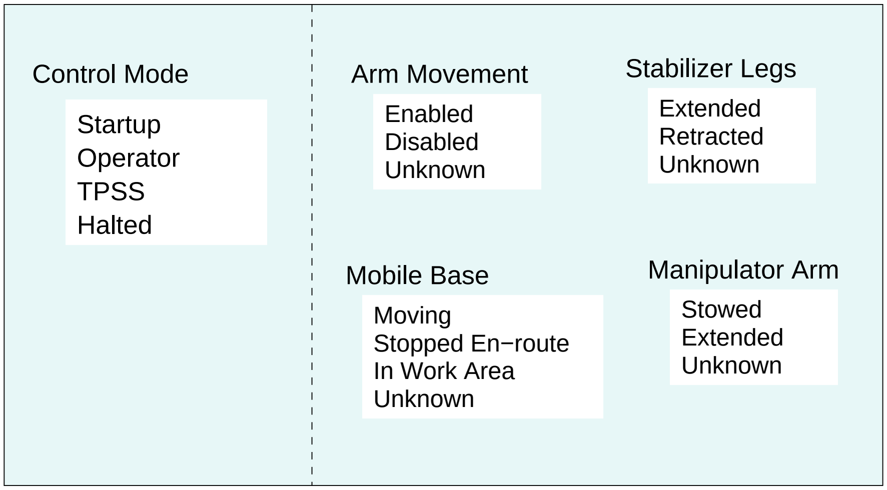


Figure 9.4: Preliminary Process Model

1. The controller must ensure the stabilizer legs are extended whenever arm movements are enabled.
2. The controller must not command a retraction of the stabilizer legs when the manipulator arm is not in a stowed position.
3. The controller must command a deployment of the stabilizer legs before arm movements are enabled; the controller must not command a retraction of the stabilizer legs before the manipulator arm is stowed.
4. The controller must not stop the leg extension until the legs are fully extended.

Similar constraints will be identified for all hazardous commands, for example, the arm controller must not enable manipulator arm movement before the stabilizer legs are completely extended.

The above system safety constraints might be enforced through physical interlocks, human procedures, etc. The STPA analysis will in the next steps provide information (1) to evaluate and compare the different design choices, (2) to design fault tolerance features, (3) to guide the test and verification procedures (or training for humans). We have used a formal behavior system modeling language, SpecTRM-RL (see Appendix D), to implement a continuous simulation environment to augment the paper analyses.

To produce detailed scenarios for the violation of safety constraints, the control structure is augmented with process models. Figure 9.4 shows a process model of the mobile robot containing the information available at this point in the system design. The preliminary design of the process models comes from the information necessary to ensure the system safety constraints hold. For example, the constraint that the arm controller must not enable manipulator movement before the stabilizer legs are completely extended implies that there must be some type of feedback to determine when the leg extension has been completed.

Either (1) a general controller model can initially be constructed and the control responsibilities allocated to individual controllers (e.g., leg controllers and arm controllers) in later steps to optimize fault tolerance and communication requirements or (2) a more preliminary allocation can be proposed and then analyzed and refined. Decisions about allocation of functionality to system components are considered as the hazard analysis and system design continues.

The control loop structure and process models are now used to identify ways the system could get into a hazardous state, i.e., the identified safety design constraints are violated. The process is driven by the classification of control flaws leading to hazards shown in Figure 7.2, that is, either the control actions or the execution of the control actions is inadequate. Each part of the control loop is evaluated with respect to the control flaw classification in Figure 7.2 and its potential to lead to the identified reasons for inadequate enforcement of the safety constraints.

As an example, consider how the process model of the state of the stabilizer legs could become inconsistent with the actual state. One possible scenario involves an external object preventing the complete extension of the stabilizer legs. In that case, the robot controller (either human or automated) may assume the stabilizer legs are extended because the extension motors have been powered up (a common type of design error). Subsequent movement of the manipulator arm would then violate the identified safety constraints.

Accidents often occur during initial system startup and when restarting after a temporary shutdown. For example, the requirements for the mobile robot might specify that it must be possible to move the mobile base out of the way in case of an emergency. If the mobile robot requires an emergency shutdown while servicing the tiles, the stabilizer legs may have to be manually retracted in order to move the robot out of the way. When the robot is restarted, the controller may assume that the stabilizer legs are still extended and arm movements may be commanded that would violate the safety constraints. Such scenarios must be prohibited in the system design.

In later stages of system development, when the logic of the controller has been specified, it can be analyzed to ensure that the identified hazardous scenarios cannot occur. The analysis results are also useful in testing and other review activities.

Additional analysis is required when humans are involved in the control loops. The analysis cannot simply consider the normative procedures because humans may not follow specified procedures. Humans will adapt their behavior to be consistent with the reality of their work environment and informal work systems will emerge as a more efficient way of attaining the conflicting goals of task performance, schedule pressures, and resource scarcity. STPA starts from the hazard, working backward to identify the types of deviations that could lead to it and can identify the types of human behavioral deviations from procedures that can lead to hazardous system states. System dynamics models can be used to assist in this process. Once these hazardous human behaviors are identified, the system design might be changed to prevent or reduce their occurrence or, if this is not possible, prevention and mitigation strategies might involve training, monitoring for safety-critical deviations, and developing operator skills at judging when, when not, and how to adapt procedures to local circumstances.

Just as operator behavior may degrade or change over time, so may other parts of the hierarchical safety control structure. An effective safety control structure at the beginning of the system life cycle may become less effective at enforcing safety constraints as a result of asynchronous evolution of the components of the structure or eroding control mechanisms. Traditional hazard analyses are static in nature, focusing on the ability of the system to avoid unsafe states given the current system design and its environment. In contrast, a STAMP-based hazard analysis assumes that systems are dynamic in nature and will evolve and adapt based on changes within the system and in its operating environment as well as economic pressures, tight schedules, and creeping expectations. A complete hazard analysis must therefore identify the possible changes to the safety control structure over time that could lead to a high-risk state. We believe system dynamics models can be useful in identifying migration of the system to states of elevated risk. Chapter 10 describes this further. This information can be used to prevent such changes through system design, or, if that is not possible, to generate operational metrics and auditing procedures to detect such degradation and to design controls or maintenance, system change, and upgrade activities.

9.3 Integrating Risk into Early System Trade Studies

(See paper on this topic in class directory)

Chapter 10

Organizational Risk Analysis and Performance Monitoring

Define risk analysis. Most done in an ad hoc way. This chapter presents a rigorous way to perform a risk analysis based on STAMP, STPA, and basic system safety principles.

Chapter 11

Process and Specifications

Safety must be designed into a system from the beginning, which requires that the designers have the information necessary to accomplish that goal. Too often today, system safety engineers are busy creating safety analyses while the system engineers are in parallel making critical decisions about system design and concepts of operations that are not based on that hazard analysis. By the time the system engineers get the information created by the system safety engineers, it is too late to have the impact on design decisions that are necessary to be most effective. If and when they get the results of the system safety activities, often in the form of a critique of the design late in the development process, the concerns are frequently ignored or argued away because changing the design at that time is too costly.

Documenting and tracking hazards and their resolution are basic requirements for any effective safety program. But simply having the safety engineer track them is not enough—information must be derived from them to inform the system engineering process and that information needs to be specified and recorded in a way that has an impact on the decisions made during system design and operations. To have such an impact, the safety-related information required by the engineers needs to be *integrated into* the environment in which safety-related engineering decisions are made. Engineers are unlikely to be able to read through volumes of hazard analysis information and relate it easily to the specific component upon which they are working. The information the system safety engineer has painfully generated must be presented to the system designers, implementers, maintainers, and operators in a way that they can easily find it when they need to make decisions based on it.

In addition to the conceptual design stages, information needs to be presented in a form that people can learn from, apply to their daily jobs, and use throughout the life cycle of projects. Too often, preventable accidents have occurred due to changes that were made after the initial design period. Accidents are frequently the result of safe designs becoming unsafe over time when changes (in the system itself or in its environment) violate the basic assumptions of the original hazard analysis. These problems are most important in complex systems where nobody is able to keep all the information necessary to make safe decisions in their head.

The key to avoiding these problems lies in specifications. While engineers may have been able to get away with minimal specifications during development of the simpler electromechanical systems of the past, specifications are critical to the successful engineering of systems with the size and complexity we are attempting to build today. Specifications are no longer simply a way of archiving information, but need to play an active role in the system engineering process.

Specification languages can help (or hinder) human performance of the various problem-solving activities involved in system requirements analysis, hazard analysis, design, review, verification and validation, debugging, maintenance and evolution (sustainment). They do this by including notations and tools that enhance our ability to reason about particular properties, construct the system and the software in it to achieve them, and validate (at each step, starting from the very beginning of system development) that the evolving system has the desired qualities. In addition, systems and particularly the software components are continually changing and evolving; they must be designed to be changeable and the specifications must support evolution without compromising the confidence in the properties that were initially verified.

This chapter describes a system engineering process with system safety integrated and embedded in it and an approach, called intent specifications, to designing system and software specifications that potentially enhances human processing and use.

11.1 An Integrated System Engineering/System Safety Process

The system engineering process becomes more critical as our systems increase in size and complexity. Important system-level properties, such as safety and security, must be built into the design of these systems from the beginning; they cannot be added on or simply measured afterward. Up-front planning and changes to the development process are needed to achieve these objectives.

The basic system engineering process is outlined in Chapter 4. This process provides a logical structure for problem solving. Briefly, first a need or problem is specified in terms of objectives that the system must satisfy and criteria that can be used to rank alternative designs (see Figure 4.3). Then a process of system synthesis takes place that results in a set of alternative designs. Each of these alternatives is analyzed and evaluated in terms of the stated objectives and design criteria, and one alternative is selected to be implemented. In practice, the process is highly iterative: The results from later stages are fed back to early stages to modify objectives, criteria, design alternatives, and so on.

Design alternatives are generated through a process of system architecture development and analysis. The system engineers break down the system into a set of subsystems, together with the functions and constraints imposed upon the individual subsystem designs, the major system interfaces, and the subsystem interface topology. These aspects are analyzed with respect to desired system performance characteristics and constraints, and the process is iterated until an acceptable system design results. The preliminary design at the end of this process must be described in sufficient detail that subsystem implementation can proceed independently. The subsystem requirements and design processes are simply subsets of the larger system engineering process.

System engineering views each system as an integrated whole even though it is composed of diverse, specialized components, which may be physical, logical (software), or human. The objective is to design subsystems that when integrated into the whole provide the most effective system possible to achieve the overall objectives. The most challenging problems in building complex systems today arise in the interfaces between components. One example is the new highly automated aircraft where most incidents and accidents have been blamed on human error, but more properly reflect difficulties in the collateral design of the aircraft, the avionics systems, the cockpit displays and controls, and the demands placed on the pilots.

Building such systems requires integrating both system safety and human factors into the basic system engineering process. Figure 11.1 shows the types of activities that need to be performed in such an integrated process and the system safety and human factors inputs and products.

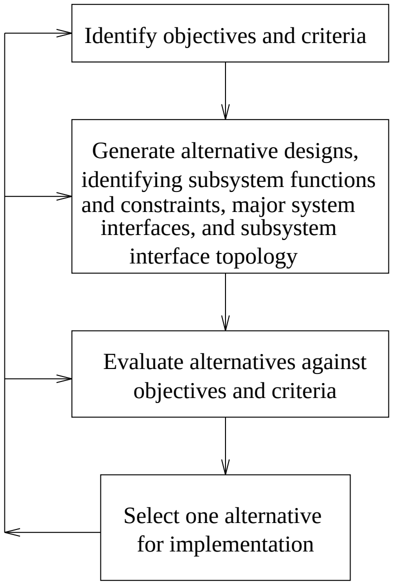

Identify objectives and criteria
Generate alternative designs, identifying subsystem functions and constraints, major system interfaces, and subsystem interface topology
Evaluate alternatives against objectives and criteria
Select one alternative for implementation

Figure 11.1: System Safety and Human Factors Aspects of the System Engineering Process

During program and project planning, a system safety plan, standards, and safety control structure need to be developed including policies, procedures, the safety management and control structure, and communication channels. An organizational risk analysis (see Chapter 10) can provide into to this process.

The system goals and environmental constraints provide input to the identification and categorization of system hazards, which along with the preliminary hazard analysis assists in identifying the safety-related system functional, operational, and organizational requirements and design constraints.

System hazard analysis should inform and guide the generation of the system conceptual design and system architecture to ensure that safety is a major consideration in the system design decisions. System hazard analysis techniques, such as STPA, can be used to generate safe design alternatives or applied to the design alternatives generated in some other way to determine if and how the system can get into hazardous states and to eliminate or control hazards in the designs.

Once the general system design concepts are agreed upon, the next step usually involves development of the system architecture and allocation of behavioral requirements to the subsystems and components. Validation of this system architecture should include a subsystem hazard analysis, perhaps including the modeling and safety analysis of the specified component behavior requirements.

The resulting subsystem and component safety requirements and design constraints (which should be traceable back to the system safety requirements and design constraints) will be used in the subsystem and component implementation and development activities and in the verification (testing and reviews).

During field testing and operations, safety change analysis, incident and accident analysis, periodic audits and performance monitoring is required to ensure that the operational system is and remains safe.

Designing safety into the system in this manner requires that the information needed for decision making is available to the right people at the right time. Specifications are critical in this process.

11.2 The Role of Specifications in Engineering Complex Systems

While the word complex is used frequently, it is rarely defined. One problem is that there are many different types or aspects of complexity and reducing it to one definition or metric oversimplifies the concept. Another problem is that complexity is a moving target—things that were complex to engineers in the 17th century might seem simple to us now given intervening advances in scientific and engineering knowledge. The latter observation provides a starting point for a general definition of complexity: a simple system has a small number of unknowns in its interactions within the system and with its environment. Systems become increasingly complex as the number of unknowns increases.

While there are many different aspects of complexity, the underlying factor in each is intellectual manageability. For example, *interactive complexity* might be defined in terms of the interactions among the system components and its environment. A system is interactively complex when the level of interactions reaches the point where they cannot be thoroughly planned, understood, anticipated, and guarded against. In interactively complex systems, designers find it difficult to consider all the potential system states and operators have difficulty handling all normal and abnormal situations and disturbances safely and effectively. Similarly, *non-linear complexity* might be defined as occurring when cause and effect are not related in an obvious way, *dynamic complexity* arises in those system aspects related to change over time, and one type of *structural complexity* occurs when the structural decomposition of the system is not consistent with the functional decomposition.

All of these aspects of complexity are related to intellectual manageability. Complexity is not simply a property of the system itself, but includes the intellectual capabilities of the designers, operators, and users of the system. Confidence can be provided about various quality factors of the systems we build by designing systems that are intellectually manageable. Unfortunately, that puts severe limits on what we can build and accomplish. An important goal of the tools we create and use in system engineering is to stretch the limits of intellectual manageability (complexity) that we can handle. This chapter considers stretching intellectual manageability from the system designers standpoint. The next one considers who to design systems that are intellectually manageable for the operators or users.

Appropriate specifications, including models and analysis tools, can stretch our current intellectual limits to allow building more complex systems with confidence they will satisfy our goals, including safety goals. The rest of this chapter describes an approach, called Intent Specifications, to achieving these goals that is grounded on psychological principles of how humans use specifications to solve problems as well as on basic systems theory and system engineering principles. Many of these principles are derived from what cognitive psychologists, engineers, and human factors experts have learned about designing and specifying human–machine interfaces. The human-machine interface provides a representation of the state of the system that the operator can use to solve problems and perform control, monitoring, and diagnosis tasks. Just as the control panel in a plant is the interface between the operator and the plant, engineering specifications are the interface between the system designers and builders and between the builders and maintainers. The specifications help the designer, builder, tester, debugger, or maintainer understand the system well enough to create a physical form or to find problems in or change the physical form.

The next three sections describe the rationale behind the design of intent specifications. Then intent specifications are defined and an extensive example shown.

11.3 Specifications and Human Problem Solving

The language in which we specify problems has an effect on our problem-solving ability and the errors we make while solving those problems. Our specification language design should reflect what is known about human limitations and capabilities.

A problem-solving activity involves achieving a goal by selecting and using strategies to move from the current state to the goal state. Success depends on selecting an effective strategy or set of strategies and obtaining the information necessary to carry out that strategy successfully. Specifications used in problem-solving tasks are constructed to provide assistance in this process. Cognitive psychology has firmly established that the representation of the problem provided to problem solvers can affect their performance (see Norman [82] for a survey of this research). In fact, Woods claims that there are no neutral representations [128]: The representations available to the problem solver either degrade or support performance. For example, problem-solving performance can be improved by providing representations that reduce the problem solver's memory load [59] and that display the critical attributes needed to solve the problem in a perceptually salient way [51].

A problem-solving strategy is an abstraction describing one consistent reasoning approach characterized by a particular mental representation and interpretation of observations [98]. Examples of strategies are hypothesis and test, pattern recognition, decision tree search, reasoning by analogy, and topological search.

The strategy selected will depend on the problem-solver's mental model. Each of the users of a specification may (and probably will) have different mental models of the system, depending on such factors as prior experience, the task for which the model is being used, and their role in the system [1, 30, 72, 100]. The same person may have multiple mental models of a system, and even having two contradictory models of the same system does not seem to constitute a problem for people [72]

Strategies also seem to be highly variable. A study that used protocol analysis to determine the trouble-shooting strategies of professional technicians working on electronic equipment found that no two sequences of actions were identical, even though the technicians were performing the same task every time (i.e., finding a faulty electronic component) [93]. Not only do search strategies vary among individuals for the same problem, but a person may vary his or her strategy dynamically during a problem-solving activity: Effective problem solvers change strategies frequently to circumvent local difficulties encountered along the solution path and to respond to new information that changes the objectives and subgoals or the mental workload needed to achieve a particular subgoal [93, 28].

It appears, therefore, that to allow for multiple users and for effective problem solving (including shifting among strategies), specifications should support all possible strategies that may be needed for a task to allow for multiple users of the representation, for shedding mental workload by shifting strategies during problem solving, and for different cognitive and problem-solving styles. Specifications need to allow users to easily find or infer the information they need regardless of their mental model or preferred problem-solving strategies. That is, the specification design should be related to the general tasks users need to perform with the information but not be limited to specific predefined ways of carrying out those tasks. The goal of specification language design should be to make it easy for users to extract and focus on the important information for the specific task at hand without assuming particular mental models or limiting the problem-solving strategies employed by the users of the document.

What types of specifications are needed to support humans in the system engineering process and to specify the results? Design decisions at each stage must be mapped into the goals and constraints they are derived to satisfy, with earlier decisions mapped (traced) to later stages of the process, resulting in a seamless and gapless record of the progression from high-level system requirements down to component requirements and designs. The specifications must also support the various types of formal and informal analysis used to decide between alternative designs and to verify the results of the design process. Finally, they must assist in the coordinated design of the components and the interfaces between them.

11.3.1 What Should Specifications Contain?

This question is critical because cognitive psychologists have determined that people tend to ignore information during problem solving that is not represented in the specification of the problem. In experiments where some problem solvers were given incomplete representations while others were not given any representation at all, those with no representation did better [35, 107]. An incomplete problem representation actually *impaired* performance because the subjects tended to rely on it as a comprehensive and truthful representation—they failed to consider important factors deliberately omitted from the representations. Thus, being provided with an incomplete problem representation (specification) can actually lead to worse performance than having no representation at all [118].

One possible explanation for these results is that some problem solvers did worse because they were unaware of important omitted information. However, both novices and experts failed to use information left out of the specifications with which they were presented, even though the experts could be expected to be aware of this information. Fischoff, who did such an experiment involving fault tree diagrams, attributed it to an “out of sight, out of mind” phenomenon [35].

One place to start in deciding what should be in a system specification is with basic systems theory, which defines a *system* as a set of components that act together as a whole to achieve some common goal, objective, or end. The components are all interrelated and are either directly or indirectly connected to each other. This concept of a system relies on the assumptions that the system goals can be defined and that systems are atomistic, that is, capable of being separated into component entities such that their interactive behavior mechanisms can be described.

The system *state* at any point in time is the set of relevant properties describing the system at that time. The system *environment* is a set of components (and their properties) that are not part of the system but whose behavior can affect the system state. The existence of a boundary between the system and its environment implicitly defines as *inputs* or *outputs* anything that crosses that boundary.

It is very important to understand that a system is always a model—an abstraction *conceived by the analyst.* For the same man-made system, an observer may see a different purpose than the designer and may also focus on different relevant properties. Thus, there may be multiple “correct” system models or specifications. To ensure consistency and enhance communication, a common specification is required that defines the:

- System boundary,

- Inputs and outputs,
- Components,
- Structure,
- Relevant interactions between components and the means by which the system retains its integrity (the behavior of the components and their effect on the overall system state), and
- Purpose or goals of the system that makes it reasonable to consider it to be a coherent entity [18].

All of these system properties need to be included in a complete system model or specification along with a description of the aspects of the environment that can affect the system state. Most of these aspects are already included in our current specification languages. But they are not enough.

One of the most important limitations of the models underlying most current specification languages, both formal and informal, is that they cannot allow us to infer what is not explicitly represented in the model, including the intention of (reason for) doing something a particular way. This limitation became very clear when some graduate students and I attempted to reverse engineer a requirements specification for the TCAS II aircraft collision avoidance system during the TCAS II certification effort. We found it impossible to derive the requirements specification strictly from the pseudocode and an accompanying English language description of the pseudocode algorithms [69]. Although the basic information was all there, the intent was largely missing and often the mapping from goals or constraints to specific design decisions. Therefore, distinguishing between requirements and artifacts of the implementation was not possible in all cases. As has been discovered by most people attempting to maintain such systems, an audit trail of the decisions and the reasons why decisions were made is absolutely essential. This was not done by TCAS over the 15 years of its development, and those responsible for the system at the time we got involved were attempting to reconstruct decision-making information from old memos and corporate memory. For the most part, only one person was able to explain why some decisions were made or why things were designed in a particular way.

Intent information is critical in the design and evolution of systems. As Harman has said, practical reasoning is concerned with what to intend while formal reasoning with what to believe [44]. “Formal logic arguments are *a priori* true or false with reference to an explicitly defined model, whereas functional reasoning deals with relationships between models, and truth depends on correspondence with the state of affairs in the real world” [44].

There is widespread agreement about the need for design rationale (intent) information in order to understand complex systems or to correctly and efficiently change or analyze the impact of changes to them. Without a record of intent, important decisions can be undone during maintenance: Many serious accidents and losses can be traced to the fact that a system did not operate as intended because of changes that were not fully coordinated or fully analyzed to determine their effects [66].

What is not so clear is the content and structure of the intent information that is needed. Simply keeping an audit trail of decisions and the reasons behind them as they are made is not practical. The number of decisions made in any large project is enormous. Even if it were possible to write them all down, finding the proper information when needed seems to be a hopeless task if not structured appropriately. What is needed is a specification of the intent—goals, assumptions, constraints, and design rationale—from the beginning, and it must be specified in a usable and perceptually salient manner that can be updated as changes are made. That is, we need a framework within which to select and specify the design decisions that are needed to develop and maintain the system.

11.3.2 The Importance of Structure in Specification Design

The *structure* of a specification is its basis for organizing information. The information may all be included somewhere, but it may be hard to find or to determine the relationship to information specified elsewhere.

Problem solving in technological systems takes place within the context of a complex causal network of relationships [27, 93, 100, 118], and those relationships need to be reflected in the specification. The information needed to solve a problem may all be included somewhere in the assorted documentation used in large projects, but it may be hard to find when needed or to determine the relationship to information specified elsewhere. Psychological experiments in problem solving find that people attend primarily to perceptually salient information [51]. The goal of specification language design should be to make it easy for users to extract and focus on the important information for the specific task at hand, which includes all potential tasks related to use of the specification.

Cognitive engineers[1] speak of this problem as "information pickup" [128]. Just because the information is in the interface does not mean that the user can find it easily. The same is true for specifications. The problem of information pickup is compounded by the fact that there is so much information in system and software specifications while only a small subset of it may be relevant in any given context.

The problem of locating required information is getting worse as system designs become more complex. A basic and often noted principle of engineering is to keep things simple. This principle, of course, is easier to state than to do. Ashby's Law of Requisite Variety [6] tells us that there is a limit to how simple we can make control systems and still have them be effective. In addition, basic human ability is not changing. If humans want to build and operate increasingly complex systems, we need to increase what is intellectually manageable. That is, we will need to find ways to *augment* human ability.

The situation is not hopeless. As Rasmussen observes, the complexity of a system is not an objective feature of the system [92]. Observed complexity depends upon the level of resolution upon which the system is being considered. A simple object becomes complex if observed through a microscope. Complexity, therefore, can only be defined with reference to a particular representation of a system, and then can only be measured relative to other systems observed at the same level of abstraction.

Thus, a way to cope with complex systems is to structure the situation such that the observer can transfer the problem being solved to a level of abstraction with less resolution. The complexity faced by the builders or users of a system is determined by their *mental models* (representations) of the internal state of the system. We build such mental models and update them based on what we observe about the system, that is, by means of our interface to the system. Therefore, the apparent complexity of a system ultimately depends upon the technology of the interface system [92], which

[1] *Cognitive engineering* is a term that has come to denote the combination of ideas from system engineering, cognitive psychology, and human factors to cope with the challenges of building high-tech systems composed of humans and machines. These challenges have necessitated augmenting traditional human factors approaches to consider the capabilities and limitations of the human element in complex systems.

includes the specification.

The solution to the complexity problem is to take advantage of the most powerful resources people have for dealing with complexity. Newman has noted, "People don't mind dealing with complexity if they have some way of controlling or handling it . . . If a person is allowed to structure a complex situation according to his perceptual and conceptual needs, sheer complexity is no bar to effective performance" [81, 92]. Thus, complexity itself is not a problem if humans are presented with meaningful information in a coherent, structured context. That leads us back to systems theory and hierarchy theory, as discussed in Chapter 4.

Two ways humans cope with complexity is to use top-down reasoning and stratified hierarchies. Building systems bottom-up works for relatively simple systems. But as the number of cases and objects that must be considered increases, this approach becomes unworkable—we go beyond the limits of human memory and logical ability to cope with the complexity. Top-down reasoning is a way of managing that complexity. At the same time, we have found that pure top-down reasoning is not adequate alone; humans need to combine top-down with bottom-up reasoning. Thus, the structure of the information must allow reasoning in both directions.

In addition, humans cope with complexity by building stratified hierarchies. Models of complex systems can be expressed in terms of a *hierarchy* of levels of organization, each more complex than the one below, where a level is characterized by having *emergent* properties and control action involves imposing constraints upon the activity at lower levels of the hierarchy (see Chapter 4). Note that describing the emergent properties resulting from the imposition of constraints requires a language at a higher level (a metalevel) *different* than that describing the components themselves.

The problem then comes down to determining appropriate types of hierarchical abstraction for system specifications that allow both top-down and bottom-up reasoning. In engineering specifications, we have made much use of *part-whole abstractions*. In such abstractions, each level of a hierarchy represents an aggregation of the components at a lower level (Figure 11.2).

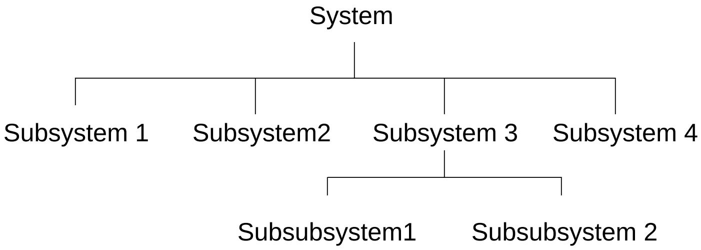


Figure 11.2: Part-Whole Abstraction

We also use *information-hiding abstractions*, where each level contains the same conceptual information but hides some details about the concepts, that is, each level is a refinement of the information at a higher level (Figure 11.3). Each refinement can be thought of as providing *what* information while the next lower level describes *how*.

Such hierarchies, however, do not provide information about *why*. Higher-level emergent information about purpose or intent cannot be inferred from what we normally include in such specifications. Design errors may result when we either guess incorrectly about higher-level intent or omit it from our decision-making process. For example, while specifying the system requirements for TCAS II [69], we learned (orally from a reviewer) that crossing maneuvers (where the aircraft paths potentially intercept if the maneuvers are not executed correctly) are avoided in the design

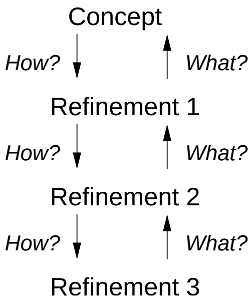


Figure 11.3: Refinement Abstraction

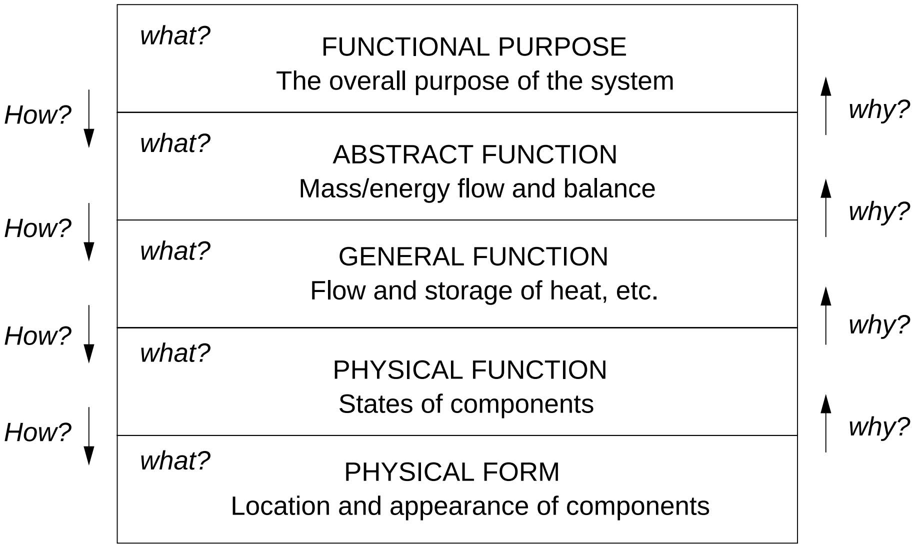


Figure 11.4: The structure of a means-ends abstraction for the design of operator interfaces.

for safety reasons. The analysis on which this decision is based comes partly from experience during TCAS system testing on real aircraft and partly as a result of an extensive safety analysis performed on the system. This design constraint would not be apparent in most design or code specifications (it was not recorded anywhere in the extensive TCAS documentation). The constraint could easily be violated during system modification unless it was documented and easily located.

There are abstractions that can be used in stratified hierarchies other than part–whole and information-hiding abstractions that overcome these limitations. While investigating the design of safe human–machine interaction, Rasmussen studied protocols recorded by people working on complex systems (process plant operators and computer maintainers) and found that they structured the system along two dimensions: (1) a part–whole abstraction in which the system is viewed as a group of related components at several levels of physical aggregation, and (2) a means–ends abstraction [93].

In a *means-ends* abstraction, each level represents a different model of the same system (Figure 11.4). At any point in the hierarchy, the information at one level acts as the goals (the ends) with respect to the model at the next lower level (the means). Thus, in a means–ends abstraction, the current level specifies *what*, the level below *how*, and the level above *why* [93]. In essence, this intent information is emergent in the sense of system theory:

> When moving from one level to the next higher level, the change in system properties represented is not merely removal of details of information on the physical or material properties. More fundamentally, information is added on higher-level principles governing the coordination of the various functions or elements at the lower level. In man-made systems, these higher-level principles are naturally derived from the purpose of the system, i.e., from the reasons for the configurations at the level considered [93]

A change of level involves both a shift in concepts and in the representation structure as well as a change in the information suitable to characterize the state of the function or operation at the various levels [93].

Each level in a means-ends hierarchy describes the system in terms of a different set of attributes or "language." Models at the lower levels are related to a specific physical implementation that can serve several purposes while those at higher levels are related to a specific purpose that can be realized by several physical implementations. Changes in goals will propagate downward through the levels while changes in the physical resources (such as faults or failures) will propagate upward. In other words, states can only be described as errors or faults with reference to their intended functional purpose. Thus reasons for proper function are derived "top-down." In contrast, causes of improper function depend upon changes in the physical world (i.e., the implementation) and thus they are explained "bottom up" [118].

Mappings between levels are many-to-many: Components of the lower levels can serve several purposes while purposes at a higher level may be realized using several components of the lower-level model. These goal-oriented links between levels can be followed in either direction, reflecting either the means by which a function or goal can be accomplished (a link to the level below) or the goals or functions an object can affect (a link to the level above). So the means–ends hierarchy can be traversed in either a top-down (from ends to means) or a bottom-up (from means to ends) direction.

As stated earlier, our representations of problems have an important effect on our problem-solving ability and the strategies we use, and there is good reason to believe that representing the problem space as a means–ends mapping provides useful context and support for decision making and problem solving. Consideration of purpose or reason (top-down analysis in a means-ends hierarchy) has been shown to play a major role in understanding the operation of complex systems [92].

Rubin's analysis of his attempts to understand the function of a camera's shutter (as cited in [94]) provides an example of the role of intent or purpose in understanding a system. Rubin describes his mental efforts in terms of thinking of the components of the shutter in terms of their function in the whole rather than explaining how each individual part worked: How they worked was immediately clear when their function was known. Rasmussen argues that this approach has the advantage that solutions of subproblems are identifiable with respect to their place in the whole picture, and it is immediately possible to judge whether a solution is correct or not. In contrast, arguing from the behavior of the individual parts to the way they function in the whole (which is what we were trying to do in reverse engineering TCAS requirements) is much more difficult because it requires synthesis: Solutions of subproblems must be remembered in isolation, and their correctness is not immediately apparent.

Support for this argument can be found in the difficulties AI researchers have encountered when modeling the function of mechanical devices “bottom-up” from the function of the components. DeKleer and Brown found that determining the function of an electric buzzer solely from the structure and behavior of the parts requires complex reasoning [24]. Rasmussen suggests that the resulting inference process is very artificial compared to the top-down inference process guided by functional considerations as described by Rubin. “In the DeKleer-Brown model, it will be difficult to see the woods for the trees, while Rubin’s description appears to be guided by a birds-eye perspective ” [94]. This may be one reason why many large projects using object-oriented design for control software have run into problems and reverted to the traditional functional decomposition approaches.

Glaser and Chi suggest that experts and successful problem solvers tend to focus first on analyzing the functional structure of the problem at a high level of abstraction and then narrow their search for a solution by focusing on more concrete details [39]. Representations that constrain search in a way that is explicitly related to the purpose or intent for which the system is designed have been shown to be more effective than those that do not because they facilitate the type of goal-directed behavior exhibited by experts [117]. Therefore, we should be able to improve the problem solving required in system development and evolution tasks by providing a representation (i.e., specification) of the system that facilitates goal-oriented search by making explicit the goals related to each functional component.

Viewing a system from a high level of abstraction is not limited to a means–ends hierarchy, of course. Most hierarchies allow one to observe systems at a less detailed level. The difference is that the means–ends hierarchy is explicitly *goal oriented* and thus assists goal-oriented problem solving. With other hierarchies (such as those based on part–whole abstractions), the links between levels are not necessarily related to goals. So although it is possible to use higher-levels of abstraction to select a subsystem of interest and to constrain search, the subtree of the hierarchy connected to a particular subsystem does not necessarily contain system components relevant to the goals the problem solver is considering.

11.3.3 Intent Specifications

These basic ideas provide the foundation for what I call *intent specifications.* Intent specifications apply means-ends abstractions to system specifications. Figure 11.5 shows the seven levels of an intent specification.

An intent specification differs from a standard specification primarily in its structure, not its content: no extra specification is involved but simply organizing the information in a way that has been found to be most helpful in using it. Most complex systems have voluminous documentation, much of it redundant or inconsistent, and sometimes missing important information, particularly information about *why* something was done the way it was—the intent or design rationale. Trying to determine whether a change might have a negative impact on safety, if possible at all, is usually enormously expensive and often involves regenerating analyses and work that was already done but either not recorded or not easily located when needed. Intent specifications were designed to help with these problems: Design rationale, safety analysis results, and the assumptions upon which the system design and validation are based are integrated directly into the specification and its structure rather then in separate documents so the information is at hand when needed for decision-making.

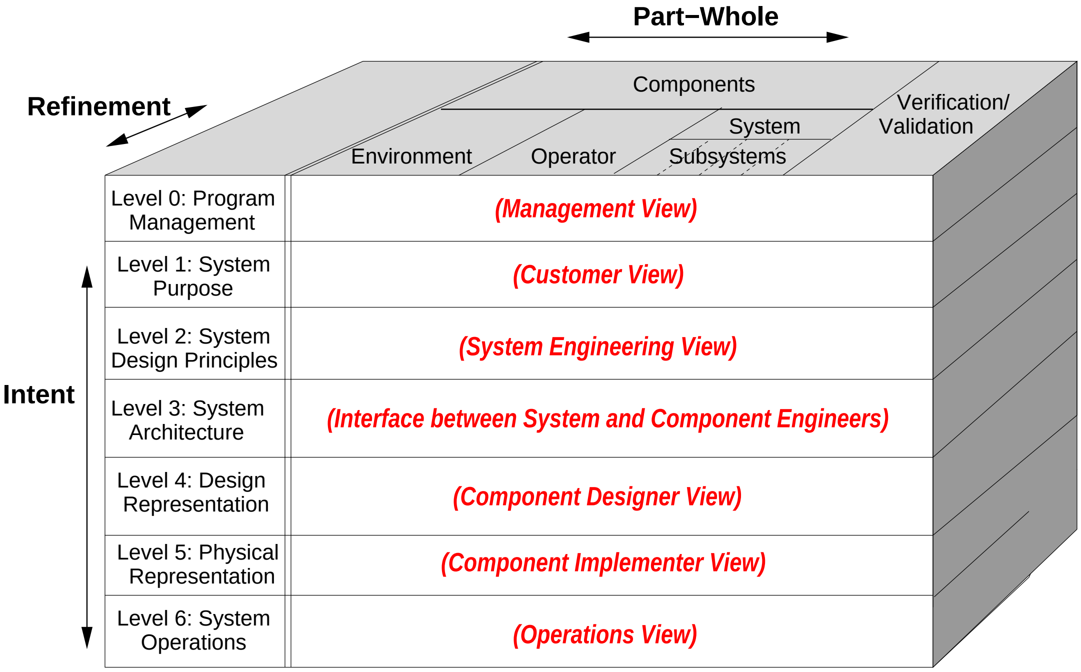

Part−Whole
Refinement
Intent
Components
Environment
Operator
System
Subsystems
Verification/ Validation
Level 0: Program Management
Level 1: System Purpose
Level 2: System Design Principles
Level 3: System Architecture
Level 4: Design Representation
Level 5: Physical Representation
Level 6: System Operations
(Management View)
(Customer View)
(System Engineering View)
(Interface between System and Component Engineers)
(Component Designer View)
(Component Implementer View)
(Operations View)

Figure 11.5: The structure of an intent specification.

Intent specifications are organized along three dimensions: intent abstraction, part-whole abstraction, and refinement (see Figure 11.5). These dimensions constitute the problem space in which the human navigates. The part-whole and refinement dimensions allow users to change their focus of attention to more or less detailed views within each level or model. The vertical dimension specifies the level of intent at which the problem is being considered, i.e., the language or model that is currently being used.

Along the horizontal part-whole dimension, intent specifications are broken up into several parts: (1) Information about characteristics of the environment that affects the ability to achieve the system goals and design constraints; (2) information about human operators or users to assist in designing "human-centered automation"; (3) information about the system and its decomposition into physical and functional components, including safety-related information. Each level also contains the requirements for and results of verification and validation activities for that level, including hazard analysis. As shown in the example in the next section, the safety information is embedded in each level (instead of being maintained in a separate safety log) but linked together so that it can easily be located.

The vertical intent dimension has seven levels, representing means-ends abstractions rather than the more usual refinement abstractions. Each level represents a different model or view of the system from a different perspective and supports a different type of reasoning about it. Refinement and decomposition occurs within each level of the specification, rather than between levels. Each level provides information not just about *what* and *how*, but *why*, that is, the design rationale and reasons behind the design decisions, including safety considerations.

The top level (Level 0) provides a project management view and insight into the relationship between the plans and the project development status through links to the other parts of the intent specification.

Level 1 is the customer view and assists system engineers and customers in agreeing on what should be built and, later, whether that has been accomplished.

Level 2 is the system engineering view and helps engineers to record and reason about the system in terms of the physical principles and system-level design principles upon which the system design is based.

Level 3 specifies the system architecture and serves as an unambiguous interface between system engineers and component engineers or contractors. At level 3, the system functions defined at Level 2 are decomposed, allocated to components, and specified rigorously and completely. Black box behavioral component models are used to specify and reason about the logical design of the system as a whole and the interactions among individual system components without being distracted by implementation details. We have found these models to be extremely helpful in engineering reviews, analysis, and simulation.

If the language used at level 3 is formal (rigorously defined), then it can play an important role in system validation. For example, the models can be executed in system simulation environments to locate system-level design errors early in the development and can be used to automate the generation of system and component test data, various types of mathematical analyses, etc. We have designed a language called SpecTRM-RL for use at Level 3 to accomplish these goals (Appendix D). Tools currently exist to create and validate the models; check for consistency, completeness, and robustness; check for the potential to cause mode confusion in users of the automation; execute the models in a simulation environment; and support STPA. Tools to create alternative visualizations of the information in the model assist in dealing with very large and complex systems [29].

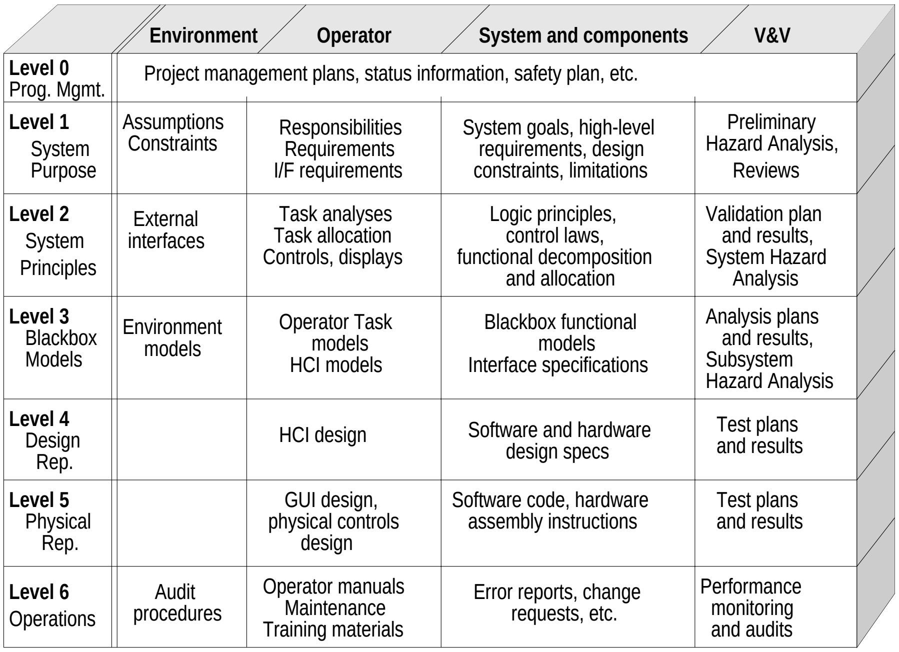
The next two levels, Design Representation and Physical Representation, provide the information necessary to reason about individual component design and implementation issues. Some parts of Level 4 may not be needed if at least portions of the physical design can be generated automatically from the models at Level 3 (as is true for software).

The final level, Operations, provides a view of the operational system and is useful in mapping between the designed system and its underlying assumptions about the operating environment envisioned during design and the actual operating environment. It assists in designing and performing system safety activities during system operations.

Each level of an intent specification supports a different type of reasoning about the system, with the highest level assisting systems engineers in their reasoning about system-level goals, constraints, priorities, and tradeoffs. The second level, System Design Principles, allows engineers to reason about the system in terms of the physical principles and laws upon which the design is based. The Architecture level enhances reasoning about the logical design of the system as a whole and the interactions between the components as well as the functional state without being distracted by implementation issues. The next two levels provide the information necessary to reason about individual component design and implementation issues.

Mappings (implemented by hyperlinks) between levels provide the relational information that allows reasoning across hierarchical levels and traceability of requirements to design. Hyperlinks are used both within and between levels. These mappings provide the relational information that allows reasoning within and across levels, including the tracing from high-level requirements down to implementation and vice versa.

The structure of the specification does not imply that the development must proceed from the top levels down to the bottom levels in that order, only that at the end of the development process,



	Environment	Operator	System and components	V&V
Level 0 Prog. Mgmt.	Project management plans, status information, safety plan, etc.			
Level 1 System Purpose	Assumptions Constraints	Responsibilities Requirements I/F requirements	System goals, high-level requirements, design constraints, limitations	Preliminary Hazard Analysis, Reviews
Level 2 System Principles	External interfaces	Task analyses Task allocation Controls, displays	Logic principles, control laws, functional decomposition and allocation	Validation plan and results, System Hazard Analysis
Level 3 Blackbox Models	Environment models	Operator Task models HCI models	Blackbox functional models Interface specifications	Analysis plans and results, Subsystem Hazard Analysis
Level 4 Design Rep.		HCI design	Software and hardware design specs	Test plans and results
Level 5 Physical Rep.		GUI design, physical controls design	Software code, hardware assembly instructions	Test plans and results
Level 6 Operations	Audit procedures	Operator manuals Maintenance Training materials	Error reports, change requests, etc.	Performance monitoring and audits

Figure 11.6: The structure of an intent specification.

all levels are complete. Almost all development involves work at all of the levels at the same time. When the system changes, the environment in which the system operates changes, or components are reused in a different system, a new or updated safety analysis is required. Intent specifications are designed to make that process feasible.

One of the advantages of using intent specifications lies not only in the capture of domain knowledge but in the potential reuse of that knowledge and of the system engineering decisions and artifacts embodying that knowledge. For example, Weiss, Ong, and Leveson have shown how generic intent specifications, particularly libraries of reusable SpecTRM-RL component models, can be reused in similar projects (see [125]).

11.3.4 Example Intent Specification

Figure 11.6 shows an example of what might be included in an intent specification. The specific information needed, of course, will vary with the type and scope of the project, but each level answers the question "why" for the design decisions in the level below. TCAS II (an airborne collision avoidance system) is used as the example throughout this section. Figure 11.7 shows the information included in our example specification.[2] Because the intent specification was created after TCAS had been developed, no management information is included. The need for Level 6 was identified only after the TCAS specification was completed and therefore it also is omitted.

In the intent specifications we have built for real systems, we have found the approach to be practical; in fact, most of the information in an intent specification is already located somewhere in

[2] The example TCAS intent specification can be found at: http://sunnyday.mit.edu/papers/intent.ps. Please note that this example was done in a limited time by the author and with limited resources. It is incomplete and almost surely differs significantly from the real TCAS system.

the often voluminous documentation for large systems. The problem in these systems usually lies in finding specific information when it is needed, in tracing the relationships between information, and in understanding the overall system design and why it was designed that way.

Management View of the Project (Level 0)

One problem in managing large projects is simply getting visibility into the progress of the project, particularly when a lot of software is involved. The highest level of an intent specification is the project management view. Here one might put project plans, such as risk management plans, pert charts, etc. along with status and pointers to the location of detailed information about each particular aspect of project development. The system safety plan will reside at this level, with embedded links to the various parts of the intent specification that implement the parts of the plan, such as the hazard list, various hazard analysis results, etc.

System Purpose or Customer View (Level 1)

Level 1 is the customer's view of the project and includes the contractual requirements (*shall* statements) and design constraints. Depending on the project, it might also contain the system goals; high-level functional requirements; system design constraints, including safety constraints; design evaluation criteria and priorities, including definitions and severity categories to be used for accidents; hazard lists and preliminary hazard analyses; the results of analyses of other system properties such as operator task analysis; assumptions about and constraints on the operating environment; and documentation of system limitations. Assumptions and rationale for the requirements and constraints are embedded in the document, as shown in the example below.

Level 1 of our TCAS intent specification includes historical information on previous attempts to build collision avoidance systems and why they were unsuccessful as well as a general introduction to the problem and the approach taken for TCAS design. The environment in which TCAS will execute is described, such as the antennas it can use and the other systems on the aircraft with which TCAS must interact.

Environment requirements and constraints[3] may lead to restrictions on the use of the system or to the need for system safety and other analyses to determine that the requirements hold for the larger system in which the system being designed is to be used. Examples for TCAS include:

E1: *The behavior or interaction of non-TCAS equipment with TCAS must not degrade the performance of the TCAS equipment or the performance of the equipment with which TCAS interacts.*

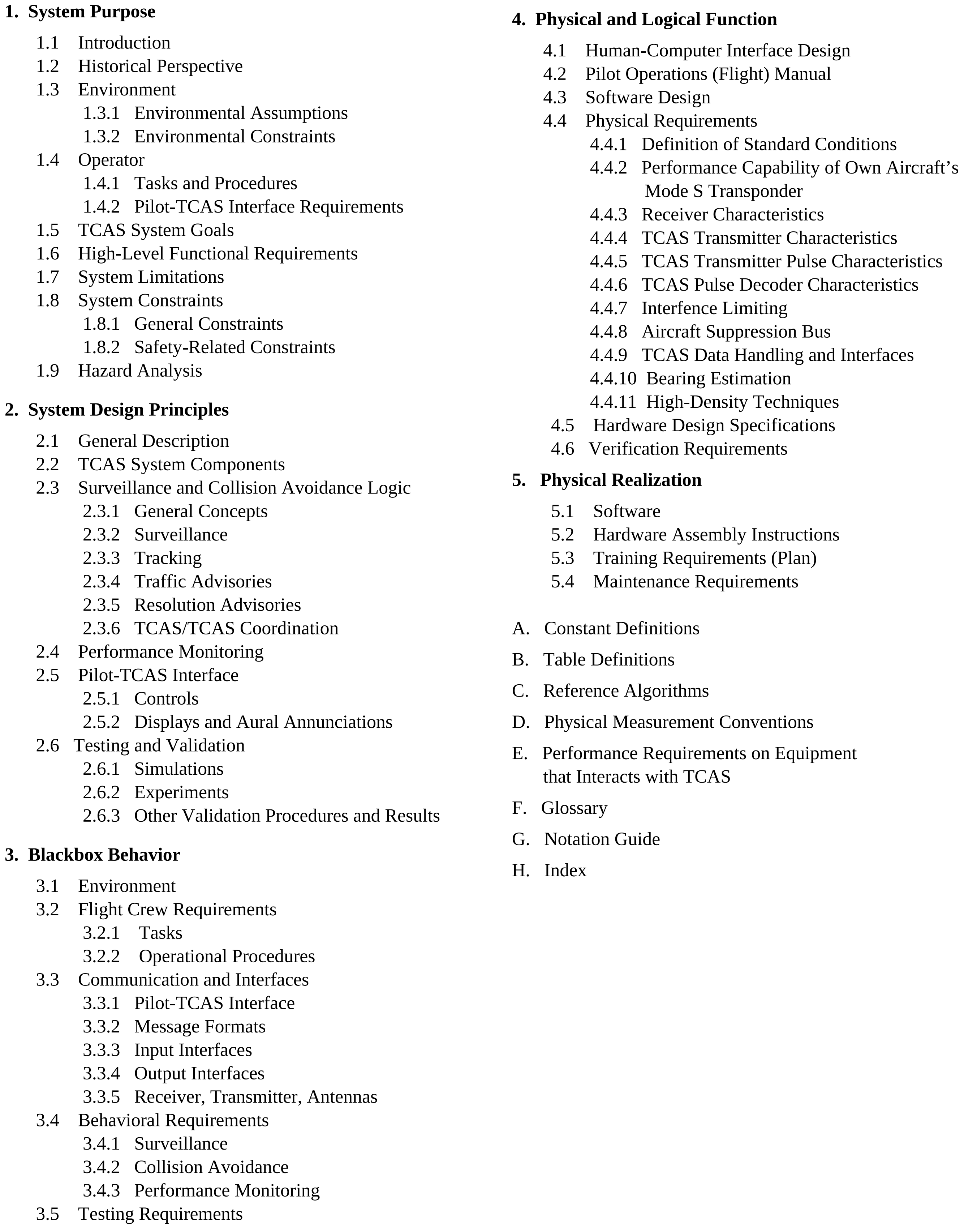
E2: *Among the aircraft environmental alerts, the hierarchy shall be: Windshear has first priority, then the Ground Proximity Warning System (GPWS), then TCAS.*

E3: *The TCAS alerts and advisories must be independent of those using the master caution and warning system.*

Assumptions may also be specified for features of the environment. Examples of environment assumptions for TCAS are that:

EA1: *High-integrity communications exist between aircraft.*

[3] Requirements and design constraints are distinguished here by using *shall* for requirements and *must* or *must not* for constraints.



1. System Purpose

- 1.1 Introduction
- 1.2 Historical Perspective
- 1.3 Environment
  - 1.3.1 Environmental Assumptions
  - 1.3.2 Environmental Constraints
- 1.4 Operator
  - 1.4.1 Tasks and Procedures
  - 1.4.2 Pilot-TCAS Interface Requirements
- 1.5 TCAS System Goals
- 1.6 High-Level Functional Requirements
- 1.7 System Limitations
- 1.8 System Constraints
  - 1.8.1 General Constraints
  - 1.8.2 Safety-Related Constraints
- 1.9 Hazard Analysis

2. System Design Principles

- 2.1 General Description
- 2.2 TCAS System Components
- 2.3 Surveillance and Collision Avoidance Logic
  - 2.3.1 General Concepts
  - 2.3.2 Surveillance
  - 2.3.3 Tracking
  - 2.3.4 Traffic Advisories
  - 2.3.5 Resolution Advisories
  - 2.3.6 TCAS/TCAS Coordination
- 2.4 Performance Monitoring
- 2.5 Pilot-TCAS Interface
  - 2.5.1 Controls
  - 2.5.2 Displays and Aural Annunciations
- 2.6 Testing and Validation
  - 2.6.1 Simulations
  - 2.6.2 Experiments
  - 2.6.3 Other Validation Procedures and Results

3. Blackbox Behavior

- 3.1 Environment
- 3.2 Flight Crew Requirements
  - 3.2.1 Tasks
  - 3.2.2 Operational Procedures
- 3.3 Communication and Interfaces
  - 3.3.1 Pilot-TCAS Interface
  - 3.3.2 Message Formats
  - 3.3.3 Input Interfaces
  - 3.3.4 Output Interfaces
  - 3.3.5 Receiver, Transmitter, Antennas
- 3.4 Behavioral Requirements
  - 3.4.1 Surveillance
  - 3.4.2 Collision Avoidance
  - 3.4.3 Performance Monitoring
- 3.5 Testing Requirements

4. Physical and Logical Function

- 4.1 Human-Computer Interface Design
- 4.2 Pilot Operations (Flight) Manual
- 4.3 Software Design
- 4.4 Physical Requirements
  - 4.4.1 Definition of Standard Conditions
  - 4.4.2 Performance Capability of Own Aircraft's Mode S Transponder
  - 4.4.3 Receiver Characteristics
  - 4.4.4 TCAS Transmitter Characteristics
  - 4.4.5 TCAS Transmitter Pulse Characteristics
  - 4.4.6 TCAS Pulse Decoder Characteristics
  - 4.4.7 Interfence Limiting
  - 4.4.8 Aircraft Suppression Bus
  - 4.4.9 TCAS Data Handling and Interfaces
  - 4.4.10 Bearing Estimation
  - 4.4.11 High-Density Techniques
- 4.5 Hardware Design Specifications
- 4.6 Verification Requirements

5. Physical Realization

- 5.1 Software
- 5.2 Hardware Assembly Instructions
- 5.3 Training Requirements (Plan)
- 5.4 Maintenance Requirements

A. Constant Definitions

B. Table Definitions

C. Reference Algorithms

D. Physical Measurement Conventions

E. Performance Requirements on Equipment that Interacts with TCAS

F. Glossary

G. Notation Guide

H. Index

Figure 11.7: The contents of the sample TCAS Intent Specification

EA2: *The TCAS-equipped aircraft carries a Mode-S air traffic control transponder, whose replies include encoded altitude when appropriately interrogated.*

EA3: *All aircraft have operating transponders.*

EA4: *All aircraft have legal identification numbers.*

EA5: *Altitude information is available from intruding targets with a minimum precision of 100 feet.*

EA6: *The altimetry system that provides own pressure altitude to the TCAS equipment will satisfy the requirements in RTCA Standard [. . .].*

EA7: *Threat aircraft will not make an abrupt maneuver that thwarts the TCAS escape maneuver.*

Examples of high-level functional *goals* (purpose) for TCAS II are to:

G1: *Provide affordable and compatible collision avoidance system options for a broad spectrum of National Airspace System users.*

G2: *Detect potential midair collisions with other aircraft in all meteorological conditions.*

Usually, in the early stages of a project, goals are stated in very general terms. One of the first steps in defining system requirements is to refine the goals into testable and achievable high-level requirements. Examples of high-level functional requirements implementing the goals for TCAS are:

1.18: *TCAS shall provide collision avoidance protection for any two aircraft closing horizontally at any rate up to 1200 knots and vertically up to 10,000 feet per minute.*

> **Assumption**: *This requirement is derived from the assumption that commercial aircraft can operate up to 600 knots and 5000 fpm during vertical climb or controlled descent (and therefore two planes can close horizontally up to 1200 knots and vertically up to 10,000 fpm).*

1.19.1: *TCAS shall operate in enroute and terminal areas with traffic densities up to 0.3 aircraft per square nautical miles (i.e., 24 aircraft within 5 nmi).*

> **Assumption**: *Traffic density may increase to this level by 1990, and this will be the maximum density over the next 20 years.*

Assumptions are specified, when appropriate, at all levels of the intent specification to explain a decision or to record fundamental information on which the design is based. These assumptions are often used in the safety or other analyses or in making lower level design decisions. For example, operational safety depends on the accuracy of the assumptions and models underlying the design and hazard analysis processes. The operational system should be monitored to ensure (1) that it is constructed, operated, and maintained in the manner assumed by the designers, (2) that the models and assumptions used during initial decision making and design are correct, and (3) that the models and assumptions are not violated by changes in the system, such as workarounds or unauthorized changes in procedures, or by changes in the environment [66]. Operational feedback on trends, incidents, and accidents should trigger reanalysis when appropriate. Linking the assumptions throughout the document with the hazard analysis (for example, to particular boxes in the system fault trees) will assist in performing safety maintenance activities. Consider the above requirement labeled 1.18. In the future, if aircraft design changes or there are proposed changes in airspace management, the origin of the specific numbers in the requirement (1200 and 10,000) can be determined and evaluated for their continued relevance. In the absence of the documentation of such assumptions, numbers tend to become "magic" and everyone is afraid to change them.

Requirements and constraints are also included for the human operator, for the human-computer interface, and for the environment in which TCAS will operate. Requirements on the operator (in this case, the pilot) are used to guide the design of the TCAS-pilot interface, the design of the automation logic, flightcrew tasks and procedures, aircraft flight manuals, and training plans and programs. Links are provided to show the relationships. Links would also be provided to parts of the hazard analysis from which safety-related requirements are derived. Example TCAS II operator requirements are:

OP.4: *After the threat is resolved, the pilot shall return promptly and smoothly to his/her previously assigned flight path (→ FTA-560, ↓3.3).*

OP.9: *The pilot must not maneuver on the basis of a Traffic Advisory only (→FTA-630, ↓2.71.3).*

Note the links to the fault tree analysis (FTA) to provide the reason for the requirement and the links to lower levels of the intent specification to show where the requirements are applied.

Design constraints are restrictions on how the system can achieve its purpose. For example, TCAS is not allowed to interfere with the ground-level air traffic control system while it is trying to maintain adequate separation between aircraft. Avoiding interference is not a goal or purpose of TCAS—the best way to achieve it is not to build the system at all. It is instead a constraint on how the system can achieve its purpose, i.e., a constraint on the potential system designs. Because of the need to evaluate and clarify tradeoffs among alternative designs, separating these two types of intent information (mission requirements and design constraints) is important.

For safety-critical systems, constraints should be further separated into normal and safety-related. Examples of non-safety constraints for TCAS II are:

C.1: *The system must use the transponders routinely carried by aircraft for ground ATC purposes (↓2.3, 2.6).*

C.4: *TCAS must comply with all applicable FAA and FCC policies, rules, and philosophies (↓2.30, 2.79).*

Safety-related constraints should have two-way links to the system hazard log and perhaps links to any analysis results that led to that constraint being identified as well as links to the design features (usually Level 2) designed to eliminate or control them. Hazard analyses are linked to Level 1 requirements and constraints, to design features on Level 2, and to system limitations (or accepted risks). Figure 11.8 shows how each leaf node of a fault tree might be linked to a safety constraint or safety-related requirement, design feature to mitigate the problem, system limitation, etc. Figure 11.9 shows a different part of the TCAS fault tree that includes operator errors and the hyperlinks point to operator procedural requirements (which in turn link to operator procedure specifications and operator manuals).

The hazard list for TCAS was shown in Section 9.1. An example of a Level 1 safety constraint derived to prevent hazards is:

SC.3: *TCAS must generate advisories that require as little deviation as possible from ATC clearances (↓FTA-550, 2.30).*

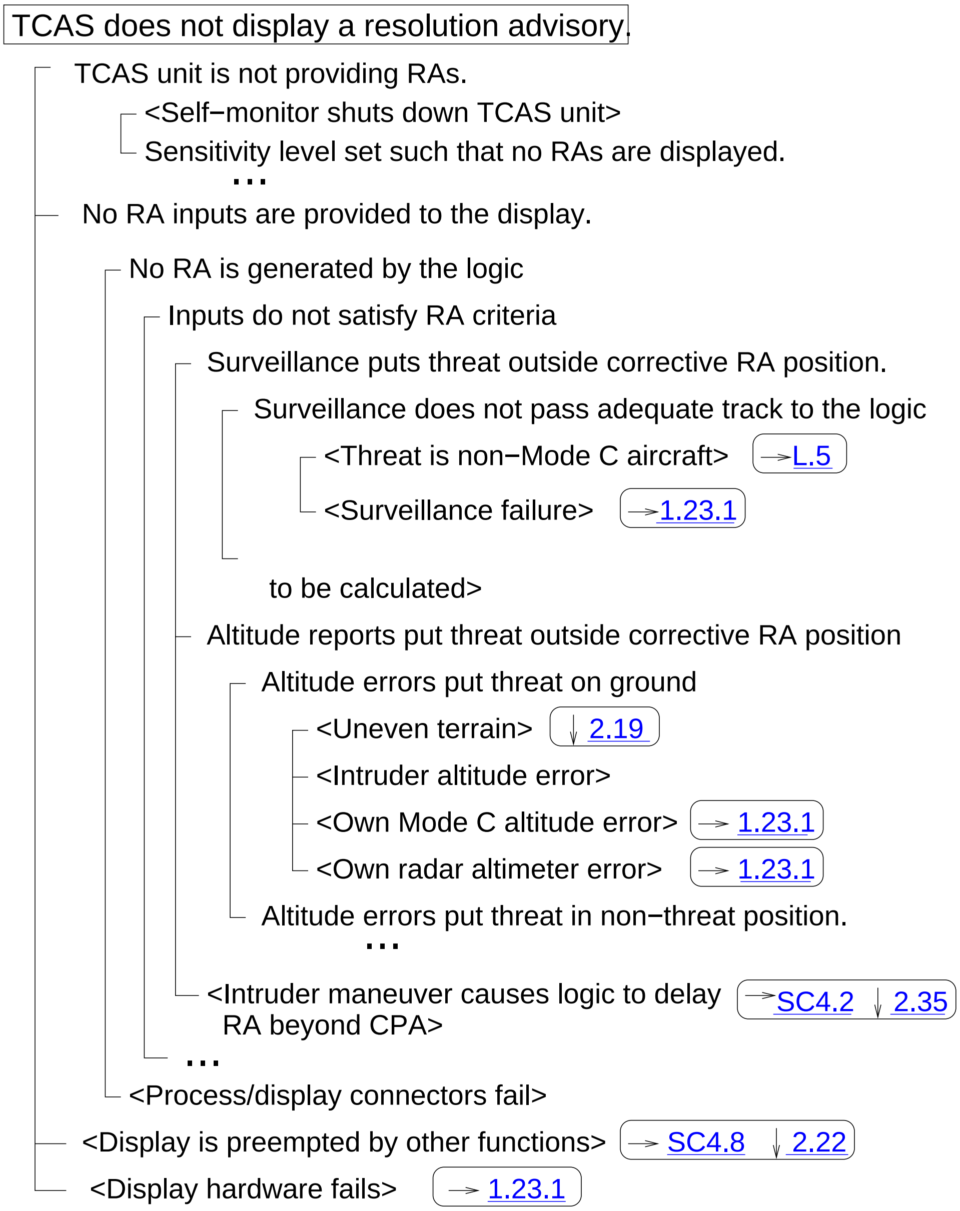

TCAS does not display a resolution advisory.
TCAS unit is not providing RAs.
<Self−monitor shuts down TCAS unit>
Sensitivity level set such that no RAs are displayed.
...
No RA inputs are provided to the display.
No RA is generated by the logic
Inputs do not satisfy RA criteria
Surveillance puts threat outside corrective RA position.
Surveillance does not pass adequate track to the logic
<Threat is non−Mode C aircraft>
→L.5
<Surveillance failure>
→1.23.1
to be calculated>
Altitude reports put threat outside corrective RA position
Altitude errors put threat on ground
<Uneven terrain>
↓ 2.19
<Intruder altitude error>
<Own Mode C altitude error>
→ 1.23.1
<Own radar altimeter error>
→ 1.23.1
Altitude errors put threat in non−threat position.
...
<Intruder maneuver causes logic to delay RA beyond CPA>
→SC4.2 ↓ 2.35
...
<Process/display connectors fail>
<Display is preempted by other functions>
→ SC4.8 ↓ 2.22
<Display hardware fails>
→ 1.23.1

Figure 11.8: Part of a Fault Tree with Hyperlinks on the Leaf Nodes

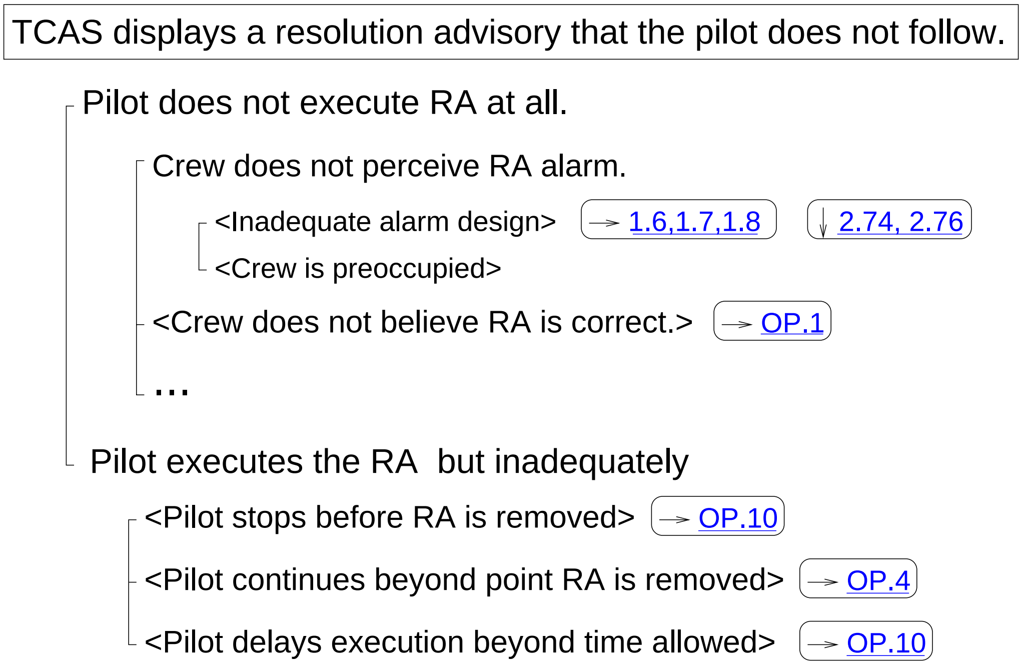

TCAS displays a resolution advisory that the pilot does not follow.
Pilot does not execute RA at all.
Crew does not perceive RA alarm.
<Inadequate alarm design>
→ 1.6,1.7,1.8
↓ 2.74, 2.76
<Crew is preoccupied>
<Crew does not believe RA is correct.>
→ OP.1
...
Pilot executes the RA but inadequately
<Pilot stops before RA is removed>
→ OP.10
<Pilot continues beyond point RA is removed>
→ OP.4
<Pilot delays execution beyond time allowed>
→ OP.10

Figure 11.9: Part of a Fault Tree Showing Links to Operator Requirements

The link to Level 2 (↓2.30) points to the system design feature on Level 2 of the intent specification that implements this safety constraint.

The following safety constraint example shows how refinement occurs at the same level of the intent specification, not at the lower levels. Lower levels are models of the system from a different perspective, not just a more detailed description of the same model.

SC.2: *TCAS must not interfere with the ground ATC system or other aircraft transmissions to the ground ATC system (→ H5).*

SC.2.1: *The system design must limit interference with ground-based secondary surveillance radar, distance-measuring equipment channels, and with other radio services that operate in the 1030/1090 MHz frequency band (↓2.5.1).*

SC.2.1.1: *The design of the Mode S waveforms used by TCAS must provide compatibility with Modes A and C of the ground-based secondary surveillance radar system (↓2.6).*

SC.2.1.2: *The frequency spectrum of Mode S transmissions must be controlled to protect adjacent distance-measuring equipment channels (↓2.13).*

SC.2.1.3: *The design must ensure electromagnetic compatibility between TCAS and [. . .] [↓2.14).*

SC.2.2: *Multiple TCAS units within detection range of one another (approximately 30 nmi) must be designed to limit their own transmissions. As the number of such TCAS units within this region increases, the interrogation rate and power allocation for each of them must decrease in order to prevent undesired interference with ATC (↓2.13).*

An example of an assumption associated with a safety constraint is:

SC.6: *TCAS must not disrupt the pilot and ATC operations during critical phases of flight nor disrupt aircraft operation (→ H3, ↓2.2.3, 2.19, 2.24.2).*

SC.6.1: *The pilot of a TCAS-equipped aircraft must have the option to switch to the Traffic-Advisory-Only mode where TAs are displayed but display of resolution advisories is inhibited (↓ 2.2.3).*

Assumption: *This feature will be used during final approach to parallel runways, when two aircraft are projected to come close to each other and TCAS would call for an evasive maneuver (↓ 6.17).*

The specified assumption is critical for evaluating safety during operations. Humans tend to change their behavior over time and use automation in different ways than originally intended by the designers. Sometimes, these new uses are dangerous. The hyperlink at the end of the assumption (↓ 6.17) points to the location in Level 6 where auditing procedures for safety during operations are defined and where the procedures for auditing this assumption would be specified.

As another example of a safety constraint, consider the following constraints arising from the TCAS NMAC (near midair collision) hazard:

SC.7: *TCAS must not create near misses (result in a hazardous level of vertical separation that would not have occurred had the aircraft not carried TCAS (→ H1).*

SC.7.1: *Crossing maneuvers must be avoided if possible (↓ 2.36, ↓ 2.38, ↓ 2.48, ↓ 2.49.2).*

SC.7.2: *The reversal of a displayed advisory must be extremely rare (↓ 2.51, ↓ 2.56.3, ↓ 2.65.3, ↓ 2.66).*[4]

SC.7.3: *TCAS must not reverse an advisory if the pilot will have insufficient time to respond to the RA before the closest point of approach (four seconds or less) or if own and intruder aircraft are separated by less than 200 feet vertically when ten seconds or less remain to closest point of approach (↓ 2.52).*

Some of the Level 2 design features used to satisfy design constraints SC.7.1 and SC.7.2 are shown in the next section.

System *limitations* are specified at Level 1 of an intent specification. Some may be related to the basic functional requirements, such as:

L4: *TCAS does not currently indicate horizontal escape maneuvers and therefore does not (and is not intended to) increase horizontal separation.*

Limitations may also relate to environment assumptions. For example:

L1: *TCAS provides no protection against aircraft without transponders or with nonoperational transponders (→EA3, FTA-430).*

L6: *Aircraft performance limitations constrain the magnitude of the escape maneuver that the flight crew can safely execute in response to a resolution advisory. It is possible for these limitations to preclude a successful resolution of the conflict (→H3, ↓2.38, 2.39).*

[4] This requirement is clearly vague and untestable. Unfortunately, I could find no definition of “extremely rare” in any of the TCAS documentation to which I had access.

L4: *TCAS is dependent on the accuracy of the threat aircraft's reported altitude. Separation assurance may be degraded by errors in intruder pressure altitude as reported by the transponder of the intruder aircraft (→EA5).*

> **Assumption:** *This limitation holds for existing airspace, where many aircraft use pressure altimeters rather than GPS. As more aircraft install GPS systems with greater accuracy than current pressure altimeters, this limitation will be reduced or eliminated.*

Limitations are often associated with hazards or hazard causal factors that could not be completely eliminated or controlled in the design. Thus they represent accepted risks. For example:

L3: *TCAS will not issue an advisory if it is turned on or enabled to issue resolution advisories in the middle of a conflict (→ FTA-405).*

L5: *If only one of two aircraft is TCAS equipped while the other has only ATCRBS altitude-reporting capability, the assurance of safe separation may be reduced (→ FTA-290).*

In our TCAS intent specification, both of these system limitations have pointers to boxes in the fault tree generated during the hazard analysis of TCAS II.

Finally, limitations may be related to problems encountered or tradeoffs made during the system design process (recorded on lower levels of the intent specification). For example, TCAS has a Level 1 performance monitoring requirement that led to the inclusion of a self-test function in the system design to determine whether TCAS is operating correctly. The following system limitation relates to this self-test facility:

L9: *Use by the pilot of the self-test function in flight will inhibit TCAS operation for up to 20 seconds depending upon the number of targets being tracked. The ATC transponder will not function during some portion of the self-test sequence (↓6.52).*

Most of these system limitations will be traced down in the intent specification levels to the user documentation. In the case of an avionics system like TCAS, L9 points to the Pilot Operations (Flight) Manual on level 6. There may be links to this limitation from information about the design of the self-test function in the lower levels of the intent specification.

Evaluation criteria and *priorities* may be included in Level 1 to indicate how conflicts among goals and design constraints are to be resolved and to guide design choices at lower levels. This information has not been included in the TCAS example specification due to lakc of information about how these decisions were made during the TCAS design process.

Finally, Level 1 contains the analysis results for system-level (emergent) properties such as safety or security. For the TCAS specification, a hazard analysis was performed and is included in the Level 1 specification with embedded links to the resolution of the hazards and hazard causes. A fault tree has links from each leaf node to functional requirements, design constraints, system design features, operational procedures, and system limitations. Whenever changes are made in safety-critical systems or software (during development or during maintenance and evolution), the safety of the change needs to be evaluated. This process can be difficult and expensive. By providing links throughout the levels of the intent specification, it should be easy to assess whether a particular design decision or piece of code was based on the original safety analysis or safety-related design constraint.

System Design Principles (Level 2)

The second level of the specification contains *System Design Principles*—the basic system design and scientific and engineering principles needed to achieve the behavior specified in the top level (as well as any "derived" requirements and design features not related to the Level 1 requirements). It is at this level that the user of the intent specification can get an overview of the system design and determine why the design decisions were made.

For TCAS, this level includes such general principles as the basic *tau* concept, which is related to all the high-level alerting goals and constraints:

2.2: *Each TCAS-equipped aircraft is surrounded by a protected volume of airspace. The boundaries of this volume are shaped by the tau and DMOD criteria (↑1.20.3).*

> **2.2.1:** *TAU: In collision avoidance, time-to-go to the closest point of approach (CPA) is more important than distance-to-go to the CPA. Tau is an approximation of the time in seconds to CPA. Tau equals 3600 times the slant range in nmi, divided by the closing speed in knots.*
>
> **2.2.2:** *DMOD: If the rate of closure is very low, a target could slip in very close without crossing the tau boundaries and triggering an advisory. In order to provide added protection against a possible maneuver or speed change by either aircraft, the tau boundaries are modified (called DMOD). DMOD varies depending on own aircraft's altitude regime (→2.2.4).*

The principles are linked to the related higher level requirements, constraints, assumptions, limitations, and hazard analysis as well as linked to lower-level system design and documentation and to other information at the same level. Assumptions used in the formulation of the design principles may also be specified at this level.

For example, design principle 2.51 (related to safety constraint SC-7.2 shown in the previous section) describes how sense reversals are handled:

2.51: *Sense Reversals: (↓ Reversal-Provides-More-Separation) In most encounter situations, the resolution advisory will be maintained for the duration of an encounter with a threat aircraft (↑SC-7.2). However, under certain circumstances, it may be necessary for that sense to be reversed. For example, a conflict between two TCAS-equipped aircraft will, with very high probability, result in selection of complementary advisory senses because of the coordination protocol between the two aircraft. However, if coordination communication between the two aircraft is disrupted at a critical time of sense selection, both aircraft may choose their advisories independently (↑FTA-1300). This could possibly result in selection of incompatible senses (↑FTA-395).*

> **2.51.1:** [. . .] *[information about how incompatibilities are handled]*

Design principle 2.51 describes the conditions under which reversals of TCAS advisories can result in incompatible senses and lead to the creation of a hazard by TCAS. The pointer labeled ↑FTA-395 is to a box in the TCAS fault tree for the near-miss hazard that includes that problem. The fault tree box ↑FTA-395 in Level 1 would have a complementary pointer to section 2.51 in Level 2. The design decisions made to handle such incompatibilities are described in 2.51.1, but that part of the specification is omitted here. 2.51 also contains a hyperlink (↓Reversal-Provides-More-Separation) to the detailed Level 3 logic used to implement the design decision.

Information about the allocation of these design decisions to individual system components and the logic involved is located in Level 3, which in turn has links to the implementation of the logic in lower levels. If a change has to be made to a system component (such as a change to a software module), it is possible to trace the function computed by that module upward in the intent specification levels to determine whether the module is safety critical and if (and how) the change might affect system safety.

As another example, the TCAS design has a built-in bias against generating advisories that would result in the aircraft crossing paths (called *altitude crossing advisories.*

2.36.2: *A bias against altitude crossing RAs is also used in situations involving intruder level-offs at least 600 feet above or below the TCAS aircraft (↑SC.7.1). In such a situation, an altitude-crossing advisory is deferred if an intruder aircraft that is projected to cross own aircraft's altitude is more than 600 feet away vertically (↓ Alt_Separation_Test).*

> **Assumption:** *In most cases, the intruder will begin a level-off maneuver when it is more than 600 feet away and so should have a greatly reduced vertical rate by the time it is within 200 feet of its altitude clearance (thereby either not requiring an RA if it levels off more than* ZTHR[5] *feet away or requiring a non-crossing advisory for level-offs begun after* ZTHR *is crossed but before the 600 foot threshold is reached).*

The example above includes a pointer down to the part of the black box requirements specification (*Alt_Separation_Test*) that embodies the design principle.

As another example of the type of links that may be found between Level 2 and the levels above and below it, consider the following. TCAS II advisories may need to be inhibited because of an inadequate climb performance for the particular aircraft on which TCAS II is installed. The collision avoidance maneuvers posted as advisories (called RAs or Resolution Advisories) by TCAS II assume an aircraft's ability to safely achieve them. If it is likely they are beyond the capability of the aircraft, then TCAS must know beforehand so it can change its strategy and issue an alternative advisory. The performance characteristics are provided to TCAS through the aircraft interface (via aircraft discretes). An example design principle (related to this problem) found on Level 2 of the intent specification is:

2.39: *Because of the limited number of inputs to TCAS for aircraft performance inhibits, in some instances where inhibiting RAs would be appropriate it is not possible to do so (↑L6). In these cases, TCAS may command maneuvers that may significantly reduce stall margins or result in stall warning (↑SC9.1). Conditions where this may occur include [...]. The aircraft flight manual or flight manual supplement should provide information concerning this aspect of TCAS so that flight crews may take appropriate action (↓ [Pilot procedures on Level 3 and Aircraft Flight Manual on Level 6).*

Finally, principles may reflect tradeoffs between higher-level goals and constraints. As examples:

[5] The vertical dimension, called ZTHR, used to determine whether advisories should be issued varies from 750 to 950 feet, depending on the TCAS aircraft's altitude.

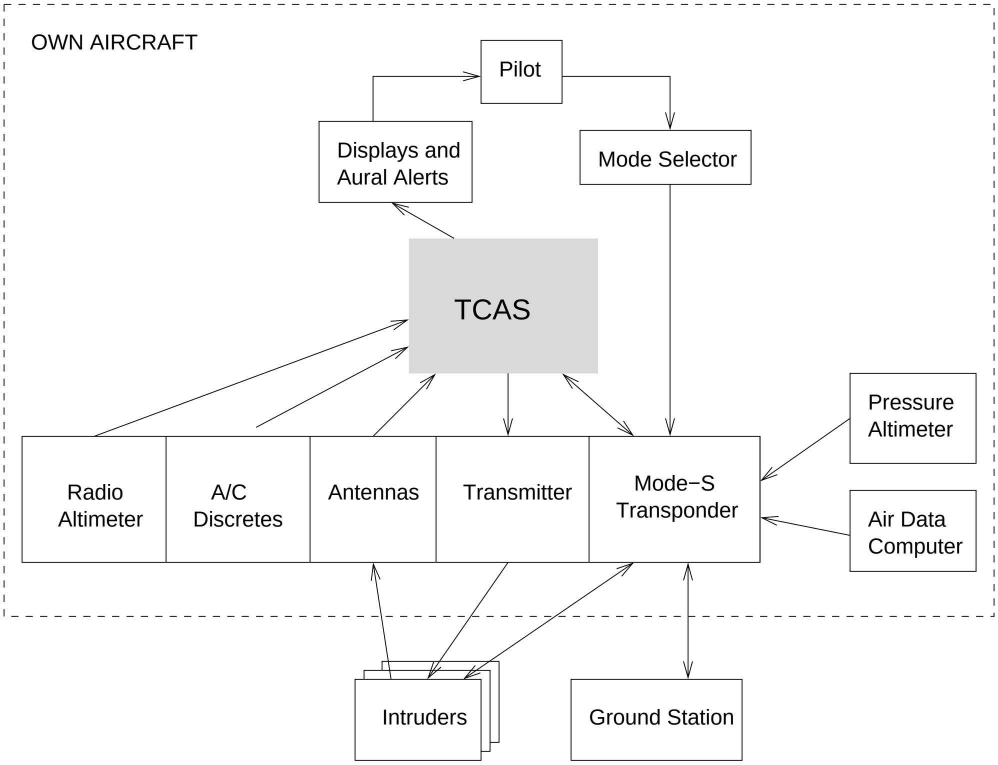


Figure 11.10: System viewpoint showing the system interface topology for the System Architecture level of the TCAS specification.

2.2.3: *Tradeoffs must be made between necessary protection (↑1.18) and unnecessary advisories (↑SC.5, SC.6). This is accomplished by controlling the sensitivity level, which controls the tau, and therefore the dimensions of the protected airspace around each TCAS-equipped aircraft. The greater the sensitivity level, the more protection is provided but the higher is the incidence of unnecessary alerts. Sensitivity level is determined by [. . .]*

2.38: *The need to inhibit* CLIMB *RAs because of inadequate aircraft climb performance will increase the likelihood of TCAS II (a) issuing crossing maneuvers, which in turn increases the possibility that an RA may be thwarted by the intruder maneuvering (↑SC7.1, FTA-1150), (b) causing an increase in* DESCEND *RAs at low altitude (↑SC8.1), and (c) providing no RAs if below the descend inhibit level (1200 feet above ground level on takeoff and 1000 feet above ground level on approach).*

System Architecture (Level 3)

Level 3 contains the System Architecture, i.e., the allocation of functions to components and the designed communication paths among those components (including human operators). SpecTRM-RL is used as the example specification language in this section (see Appendix D). Figure 11.10 shows a system-level view of TCAS II and its environment.

The environment description at Level 3 includes the assumed behavior of the external components (such as the altimeters and transponders for TCAS), including perhaps failure behavior,

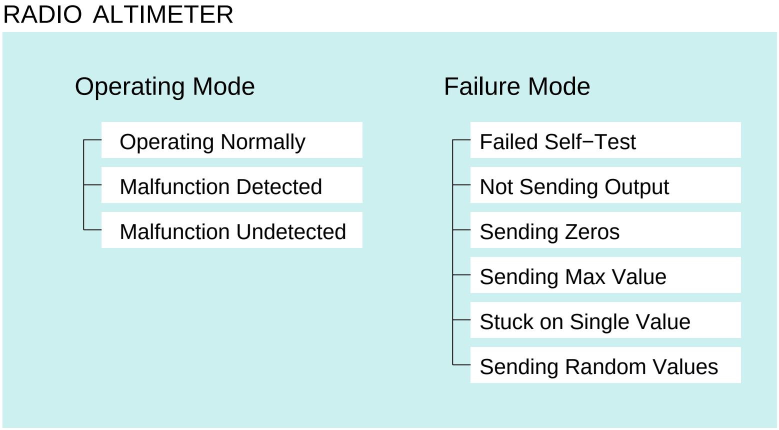


Figure 11.11: Part of the SpecTRM-RL description of an environment component (a radio altimeter). Modeling failure behavior is especially important for safety analyses. In this example, (1) the altimeter may be operating correctly, (2) it may have failed in a way that the failure can be detected by TCAS II (i.e., it fails a self-test and sends a status message to TCAS or it is not sending any output at all), or (3) the malfunctioning is undetected and it sends an incorrect radio altitude.

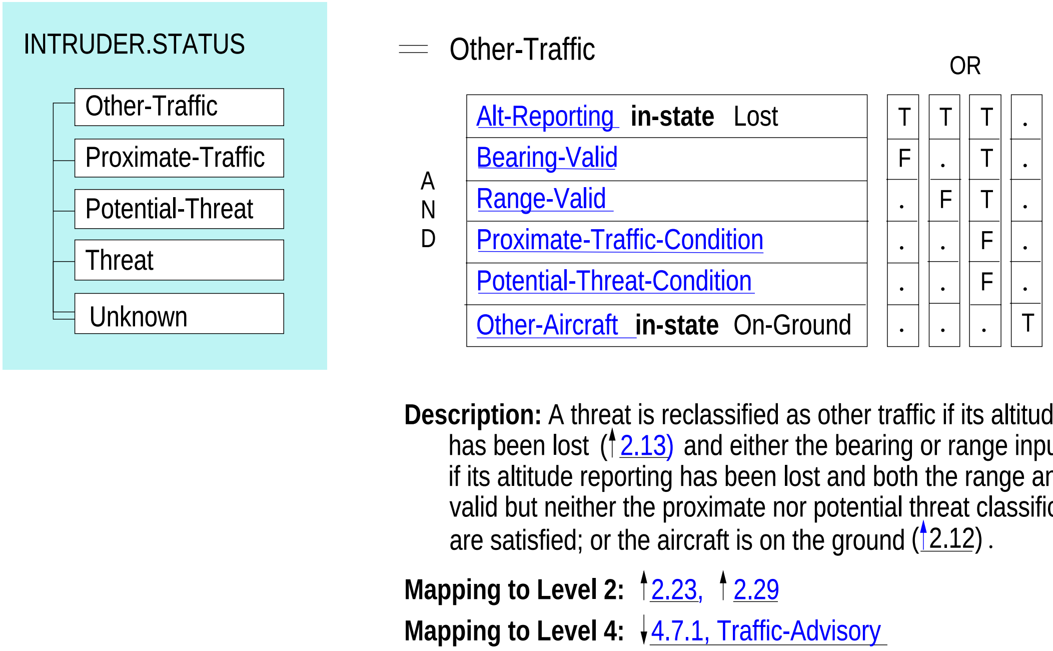
upon which the correctness of the system design is predicated, along with a description of the interfaces between the TCAS system and its environment. Figure 11.11 shows part of a SpecTRM-RL description of an environment component, in this case an altimeter.

Remember that the boundaries of a system are purely an abstraction and can be set anywhere convenient for the purposes of the specifier. In this example, the environment includes any component that was already on the aircraft or in the airspace control system and was not newly designed or built as part of the TCAS effort.

Each arrow in Figure 11.10 represents a communication and needs to be described in more detail. Each box (component) also needs to be refined. What is included in the decomposition of the component will depend on whether the component is part of the environment or part of the system being constructed. The language used to describe the components may also vary. Figure 11.12 shows part of the SpecTRM-RL description of the behavior of the CAS (collision avoidance system) subcomponent. SpecTRM-RL specifications are intended to be both easily readable with minimum instruction and formally analyzable. They are also executable and can be used in a system simulation environment. Readability was a primary goal in the design of SpecTRM-RL, as was completeness with regard to safety. Most of the requirements completeness criteria described in *Safeware* are included in the syntax of the language to assist in system safety reviews of the requirements.

SpecTRM-RL explicitly shows the process model used by the controller and describes the required behavior in terms of this model. A state machine model is used to describe the process model (the state of the aircraft and the air space around it, in this case) and the ways this model can change state.

Logical behavior is specified in SpecTRM-RL using AND/OR tables. Figure 11.12 shows a small part of the specification of the TCAS collision avoidance logic. For TCAS, an important state variable is the status of the other aircraft around the TCAS aircraft, called *intruders*. Intruders are classified into four groups: Other Traffic, Proximate Traffic, Potential Threat, and Threat. The



= Other-Traffic

OR

AND

Alt-Reporting **in-state** Lost	T	T	T	.
Bearing-Valid	F	.	T	.
Range-Valid	.	F	T	.
Proximate-Traffic-Condition	.	.	F	.
Potential-Threat-Condition	.	.	F	.
Other-Aircraft **in-state** On-Ground	.	.	.	T

Description: A threat is reclassified as other traffic if its altitude reporting has been lost (↑2.13) and either the bearing or range inputs are invalid; if its altitude reporting has been lost and both the range and bearing are valid but neither the proximate nor potential threat classification criteria are satisfied; or the aircraft is on the ground (↑2.12) .

Mapping to Level 2: ↑2.23, ↑2.29

Mapping to Level 4: ↓4.7.1, Traffic-Advisory

Figure 11.12: Example from the Level 3 SpecTRM-RL model of the collision avoidance logic. It defines the criteria for downgrading the status of an intruder (into our protected volume) from being labeled a threat to being considered simply as other traffic. Intruders can be classified in decreasing order of importance as a threat, a potential threat, proximate traffic, and other traffic. In the example, the criterion for taking the transition from state *Threat* to state *Other Traffic* is represented by an AND/OR table, which evaluates to TRUE if any of its columns evaluates to TRUE. A column is TRUE if all of its rows that have a "T" are TRUE and all of its rows with an "F" are FALSE. Rows containing a dot represent "don't care" conditions.

figure shows the logic for classifying an intruder as other traffic using an AND/OR table. We have tools to visualize this information in additional ways.

The rows of the table represent AND relationships while the columns represent OR. The state variable takes the specified value (in this case, *Other Traffic*) if any of the columns evaluate to TRUE. A column evaluates to TRUE if all the rows have the value specified for that row in the column. A dot in the table indicates that the value for the row is irrelevant. Underlined variables represent hyperlinks. For example, clicking on Alt Reporting would show how the Alt Reporting variable is defined: In our TCAS intent specification, the altitude report for an aircraft is defined as *Lost* if no valid altitude report has been received in the past six seconds. Bearing Valid, Range Valid, Proximate Traffic Condition, and Proximate Threat Condition are *macros*, which simply means they are defined using separate logic tables. The additional logic for the macros could have been inserted here, but we have found that sometimes the logic gets too complex and it is easier for specifiers and reviewers if, in those cases, the tables are broken up into smaller pieces (a form of refinement abstraction). This decision is, of course, up to the creator of the table.

Note that the behavioral descriptions at this level are purely blackbox: They describe the inputs and outputs of each component and their relationships *only* in terms of externally visible behavior. Essentially it represents the transfer function across the component. Any of these components (except the humans, of course) could be implemented either in hardware or software (and, in fact, some of the TCAS surveillance functions are implemented using analog devices by some vendors). Decisions about physical implementation, software design, internal variables, and so on are limited to levels of the specification below this one. Thus, this level serves as a rugged interface between system designers and component designers and implementers (including subcontractors).

Other information at this level might include flight crew requirements such as description of tasks and operational procedures, interface requirements, and the testing requirements for the functionality described on this level. We have developed a visual operator task modeling language that permits integrated simulation and analysis of the entire system, including human–computer interactions [8, 101].

We have found that the models at this level provide the best place to provide component reuse and build component libraries. Reuse of application software at the code level has been problematic at best, contributing to a surprising number of accidents (see Appendix B). Level 3 blackbox behavioral specifications provide a way to make the changes almost always necessary to reuse software in a format that is both reviewable and verifiable. Our Level 3 model of TCAS has been used for the past 12 years to maintain TCAS and to specify and validate changes before they are made in the various manufacturer's products. Once the changed Level 3 specifications have been validated, the links to the modules implementing the modeled behavior can be used to determine which modules need to be changed and how. Libraries of component models can also be developed and used in a plug-and-play fashion, making changes as required, in order to develop software product families [124].

Design Representation (Level 4)

The next two levels of an intent specification provide the information necessary to reason about component design and implementation. The fourth level, *Design Representation*, contains design information. Its content will depend on whether the particular function is being implemented using analog or digital devices or both. This level is the highest level of the intent specification that includes information about the physical or logical implementation of the components.

The design intent information may not all be completely linked and traceable upward to the levels above the Design Representation—for example, design decisions based on performance or other issues unrelated to requirements or constraints, such as the use of a particular graphics package because the programmers are familiar with it or it is easy to learn. Knowing that these decisions are *not* linked to higher level purpose is important during software maintenance and evolution activities.

The fourth level of the example TCAS intent specification simply contains the official pseudocode design specification (which was used by the FAA to specify the TCAS requirements), but any appropriate design specification language might be used. To assist in software code reviews and walkthroughs, the unoptimized code sections might be shown in the refinement of the Design Representation along with mappings to the actual optimized code at the lower implementation level. Reviewers often have difficulty reviewing and understanding code that has been optimized. If the modeling language used to specify the system architecture (Level 3) is formal (rigorously defined), the potential exists for automatically generating code from it.

Other information at Level 4 might include hardware design descriptions, the human–computer interface design specification, and verification requirements for the requirements and design specified on this level.

Physical Representation (Level 5)

The fifth level represents the implementer and manufacturing view and contains a description of the physical implementation of the levels above. It might include software code, hardware assembly or manufacturing instructions, engineering drawings, etc.

Operations (Level 6)

Level 6 documents the interface between development and operations. It contains performance auditing procedures, operator manuals, training materials, error reports and change requests, etc. Again, hyperlinks are used to maintain traceability, particularly traceability to design rationale and safety-critical decisions and design constraints.

11.3.5 Use of Intent Specification

As stated earlier, our representations of problems have an important effect on our problem-solving ability and the strategies we use. Intent specifications were designed to support the problem solving required to perform system, software, and safety engineering tasks, including tasks involving education and program understanding, search, design, validation, safety assurance, maintenance, and evolution.

Education and Program Understanding

Curtis *et.al.* [23] did a field study of the requirements and design process for 17 large systems. They found that substantial design effort in projects was spent coordinating a common understanding among the staff of both the application domain and of how the system should perform within it. The most successful designers understood the application domain and were adept at identifying unstated requirements, constraints, or exception conditions and mapping between these and the computational structures. This is exactly the information that is included in the higher levels of intent specifications and the mappings to the software. Using intent specifications should help with education in the most crucial aspects of the system design for both developers and maintainers and augment the abilities of both, i.e., increase the intellectual manageability of the task.

Search Strategies

Vicente and Rasmussen have noted that means-ends hierarchies constrain search in a useful way by providing traceability from the highest level goal statements down to implementations of the components [118]. By starting the search at a high level of abstraction and then deciding which part of the system is relevant to the current goals, the user can concentrate on the subtree of the hierarchy connected to the goal of interest: The parts of the system not pertinent to the function of interest can easily be ignored. This type of “zooming-in” behavior has been observed in a large number of psychological studies of expert problem solvers. Recent research on problem-solving behavior consistently shows that experts spend a great deal of their time analyzing the functional structure of a problem at a high level of abstraction before narrowing in on more concrete details [11, 17, 39, 93, 114].

With other hierarchies, the links between levels are not necessarily related to goals. So although it is possible to use higher levels of abstraction in a standard decomposition or refinement hierarchy to select a subsystem of interest and to constrain search, the subtree of the hierarchy connected to a particular subsystem does not necessarily contain system components that are relevant to the goals and constraints that the problem solver is considering.

Upward search in the hierarchy, such as that required for trouble shooting, maintenance, and evolution, is also supported by intent specifications. Vicente and Rasmussen claim (and have experimental evidence to support) that in order for operators to correctly and consistently diagnose faults, they must have access to higher-order functional information since this information provides a reference point defining how the system *should* be operating. States can only be described as errors or faults with reference to the intended purpose. Additionally, causes of improper functioning depend upon aspects of the implementation. Thus they are explained bottom up. The same argument applies to software debugging.

Minimizing the Effects of Requirements Changes

Hopefully, the highest levels of the specification will not change, but sometimes they do, especially during development as system requirements become better understood. Functional and intent aspects are represented throughout an intent specification, but in increasingly abstract and global terms at the higher levels. The highest levels represent more stable design goals that are less likely to change (such as detecting potential threats in TCAS), but when they do they have the most important (and costly) repercussions on the system and software design and development, and they may require analysis and changes at all the lower levels. We need to be able to determine the potential effects of changes and, proactively, to design to minimize them.

Reversals in TCAS are an example of this. About four years after the original TCAS specification was written, experts discovered that it did not adequately cover requirements involving the case where the pilot of an intruder aircraft does not follow his or her TCAS advisory and thus TCAS must change the advisory to its own pilot. This change in basic requirements caused extensive changes in the TCAS design, some of which introduced additional subtle problems and errors that took years to discover and rectify.

Anticipating exactly what changes will occur and designing to minimize the effects of those changes is difficult, and the penalties for being wrong are high. Intent specifications theoretically provide the flexibility and information necessary to design to ease high-level requirements changes without having to predict exactly which changes will occur: The abstraction and design are based on intent (system requirements) rather than on part-whole relationships (which are the least likely to change with respect to requirement or environment changes).

An example of a design criterion for this goal might be to minimize the number of one-to-many mappings between levels in order to constrain downward search and limit the effects of changes in higher levels upon the lower levels. Minimizing many-to-many (or many-to-one) mappings would, in addition, ease activities that require following upward links and minimize the side effects of lower-level changes.

Intent specifications assist in identifying intent-related structural dependencies (many-to-many mappings across hierarchical levels) to allow minimizing them during design, and they clarify the tradeoffs being made between conflicting goals.

Safety Assurance

A complete safety analysis and methodology for building safety-critical systems requires identifying the system-level safety requirements and constraints and then tracing them down to the components [66]. After the safety-critical behavior of each component has been determined (including the implications of its behavior when the components interact with each other), verification is required that the components do not violate the identified safety-related behavioral constraints. In addition, whenever any change is made to the system or when new information is obtained that brings the safety of the design into doubt, revalidation is required to ensure that the change does not degrade system safety. To make this verification (and reverification) easier, safety-critical parts of the software should be isolated and minimized.

In addition, whenever any change is made to the system, revalidation is required to ensure that the change does not affect system safety. Many accidents can be traced to the fact that the system did not operate as intended because of changes that were not fully coordinated or fully analyzed to determine their effect on the system [66]. Changes can inadvertently eliminate important safety features or diminish the effectiveness of hazard controls. For this reason, re-analysis of the design's safety features must always be performed when some known change occurs or when new information is obtained that brings the safety of the design into doubt.

This analysis cannot be performed efficiently unless those making decisions about changes and those actually making the changes know which parts of the system affect a particular safety design constraint. Specifications need to include a record of the design decisions related to basic safety-related system goals, constraints, and hazards (including both general design principles and criteria and detailed design decisions), the assumptions underlying these decisions, and why the decisions were made and particular design features included. Intent specifications capture this information and provide the ability to trace design features upward to specific high-level system goals and constraints.

Software Maintenance and Evolution

Although intent specifications provide support for a top-down, rational design process, they may be even more important for the maintenance and evolution process than for the original designer. Software evolution is challenging because it involves many cognitive processes that require complex problem-solving strategies—such as understanding the system's structure and function, understanding the code and documentation and the mapping between the two, and locating inconsistencies and errors.

Intent specifications provide the structure required for recording the most important design rationale information, i.e., that related to the purpose and intent of the system, and locating it when needed. They, therefore, can assist in the software change process.

While trying to build a model of TCAS, we discovered that the original conceptual model of the TCAS system design had degraded over the years as changes were made to the pseudocode to respond to errors found, new requirements, better understanding of the problem being solved, enhancements of various kinds, and errors introduced during previous changes. The specific changes made often simplified the process of making the change or minimized the amount of code that needed to be changed, but complicated or degraded the original model. Not having any clear representation of the model also contributed to its degradation over the ten years of changes to the pseudocode.

By the time we tried to build a representation of the underlying conceptual model, we found that the system design was unnecessarily complex and lacked conceptual coherency in many respects, but we had to match what was actually flying on aircraft. I believe that making changes without introducing errors or unnecessarily complicating the resulting conceptual model would have been simplified if the TCAS staff had had a blackbox requirements specification of the system. Evolution of the pseudocode would have been enhanced even more if the extra intent information had been specified or organized in a way that it could easily be found and traced to the code. In fact, the specification of TCAS II we provided to the FAA in 1992 (which included only Level 3 using a modeling language that was a predecessor of SpecTRM-RL) is still used to evaluate potential changes to TCAS before they are provided to the TCAS equipment manufacturers.

Reuse and Component-Based System Engineering

Chapter 12

Human-Centered Automation Design (Human-Computer Interaction)

Chapter 13

Safety Culture and Management

[*I have just started writing this chapter, and it is still very rough and incomplete. Missing or incomplete sections will be covered in the class and the class notes.*]

Many of the ideas in this section come from discussions and papers with my colleagues at the MIT Sloan School of Management, Engineering Systems Division, including Joel Cutcher-Gershenfeld, John Carroll, and Betty Barrett.

Safety culture has always been a critical part of achieving safety goals, but its importance has recently been underlined in the Columbia Accident Investigation Board Report:

> The foam debris hit was not the single cause of the *Columbia* accident, just as the failure of the joint seal that permitted O-ring erosion was not the single cause of *Challenger*. Both *Columbia* and *Challenger* were lost also because of the failure of NASA's organizational system [38, p. 195].

Perhaps the most important finding of the report was the insistence that NASA go beyond analysis of the immediate incident to address the "political, budgetary, and policy decisions" that impacted the Space Shuttle Program's "structure, culture, and safety system," which was, ultimately, responsible for flawed decision-making.

What is safety culture? A culture is a shared set of norms and values, a way of looking at and interpreting the world and events around us (our mental model) and taking action in a social context. Safety culture is the subset of culture that reflects the general attitude and approaches to safety and risk management.

Schein argues that organization culture can be divided into three tiers: (1) surface-level cultural artifacts (the routine aspects of everyday practice); (2) a middle level of stated organizational rules, values, and practices; and (3) an often invisible, but pervasive, underlying level of deep cultural operating assumptions under which actions are taken and decisions are made [103]. Culture change is difficult, Schein notes, because it requires change at all three levels and each is progressively harder to address.

Management, resources, capabilities, and culture are intertwined, and trying to change the culture without changing the environment within which the culture operates is doomed to failure. At the same time, simply changing the organizational structures (including policies, goals, missions, job descriptions, and standard operating procedures related to safety) may lower risk over the short term, but superficial fixes that do not address the set of shared values and social norms are very likely to be undone over time. As an example, the changes and protections instituted at NASA after the Challenger accident degraded over time to the point where the same performance pressures and unrealistic expectations implicated in the Challenger loss contributed also to the Columbia loss. To achieve lasting results requires making broad changes that provide protection from and appropriate responses to the continuing environmental influences and pressures that tend to degrade the safety culture over time.

A common cultural flaw found in accident investigations is a *culture of denial*, where risk assessment is unrealistic and where credible risks and warnings are dismissed without appropriate investigation. Such a culture of denial is common where embedded, operational assumptions do not match the stated organizational policies. To "engineer" a safety culture, or, in other words, to bring the operational practices and values into alignment with the stated safety values requires first identifying the desired organizational safety principles and values and then establishing and engineering the organizational infrastructure to achieve those values and to sustain them over time. "Sloganeering" is not enough—all aspects of the culture that affect safety must be engineered to be in alignment with the organizational safety principles. Successfully achieving this alignment process requires understanding why an organization's operational practices have deviated from the stated principles and not only making the appropriate adjustments but instituting protections against future misalignments.

The following are all important social system aspects of a strong safety culture:

- The *formal organizational structure* including the safety groups as well as the formal safety roles and responsibilities of executives, managers, engineers, civil servants, union leaders, and others. This formal structure is usually not static but rather a dynamic, constantly evolving set of formal relationships.

- *Organizational subsystems* that impact the safety culture and risk management including open and multi-directional communication systems; safety information systems to support planning, analysis and decision making; reward and reinforcement systems; selection and retention systems that promote safety knowledge, skills, and ability; learning and feedback systems from incidents, operational anomalies, and other aspects of operational experience; and channels and procedures for expressing safety concerns and resolving conflicts. Both formal and informal communication channels are important as well as the safety information system, which should include at least the hazard logs, problem reports, lessons learned, safety metrics, and root cause analyses. The reward and reinforcement systems should support attention to safety and not create conflicting incentives, such as rewards for schedule performance that risk compromising safety. Selection and retention systems are relevant to safety with respect to the skill sets and mindsets that are emphasized in hiring, as well as the knowledge and skills that are lost through retirements and other forms of turnover. Learning and feedback systems are central to the development and sustainment of safety knowledge and capability, while complaint and conflict resolution systems provide an essential feedback loop (including support for periodic whistle-blowing actions).

- *Informal organizational structures and social interaction processes*, including leadership, negotiations, problem solving, decision-making, and partnership. Here the focus is on leadership and decision making about safety matters at every level of the organization. Problem solving after incidents and operational anomalies is an important component of the safety culture, particularly as it relates to identifying and eliminating root causes rather than merely the symptoms of the deeper problems.

- *Individual capability and motivation*, including knowledge, skills, and ability; group dynamics, and many psychological factors including fear of surfacing safety concerns, learning from mistakes without blame, commitment to safety values, and so on.

- *Safety rules and procedures* along with their underlying values and assumptions and a clearly expressed system safety vision. The vision must be shared among all the organization's members, not just articulated by the leaders. There is often great variance in the degree to which safety visions expressed by leaders are shared among all the participants.

13.0.6 Organizational Structure

The organizational structure includes the formal organizational chart, various operating structures (such as integrated product and process design teams), various formal and informal networks, institutional arrangements, and other elements. As organizational change experts have long known, structure drives behavior.

When determining the most appropriate placement for safety activities within the organizational structure, some basic principles should be kept in mind, including:

- System safety needs a direct link to decision makers and influence on decision making;
- System safety needs to have independence from project management (but not engineering);
- Direct communication channels are needed to most parts of the organization.

These structural principles serve to ensure that system safety is in a position where it can obtain information directly from a wide variety of sources so that information is received in a timely manner and without filtering by groups with potential conflicting interests. The safety activities also must have focus and coordination. Although safety issues permeate every part of the development and operation of a complex system, a common methodology and approach will strengthen the individual disciplines. Communication is also important because safety motivated changes in one subsystem may affect other subsystems and the system as a whole. Finally, it is important that System Safety efforts do not end up fragmented and uncoordinated. While one could argue that safety staff support should be integrated into one unit rather than scattered in several places, an equally valid argument could be made for the advantages of distribution. If the effort is distributed, however, a clear focus and coordinating body are needed. A case can be made that centralization of system safety in a quality assurance organization (matrixed to other parts of the organization) that is neither fully independent nor sufficiently influential was a major factor in the decline of the safety culture at NASA preceding the Columbia loss.

Influence and Prestige of Safety Function: It is important to first recognize that there are many aspects of system safety and that putting them all into one organization may be a mistake. Safety concerns span the life cycle and safety should be involved in just about every aspect of development and operations. Safety concerns are an integral part of most engineering activities.

Putting all of the safety engineering activities into the quality assurance organization with a weak matrix structure that provides safety expertise to the projects has set up the expectation that system safety is an after-the-fact or auditing activity only. In fact, the most important aspects of system safety involve core engineering activities such as building safety into the basic design and proactively eliminating or mitigating hazards. By treating safety as an assurance activity only, safety concerns are guaranteed to come too late in the process to have an impact on the critical design decisions.

Beyond associating safety only with assurance, placing it in an assurance group can have a negative impact on its stature and thus influence. Assurance groups often do not have the prestige necessary to have the influence on decision making that safety requires.

Independence of the Safety Function: Ironically, organizational changes made after the Challenger accident in order to increase independence of safety activities had the opposite result. The project manager decided how much safety was to be "purchased" from this separate function. Therefore, as noted in the CAIB report, the very livelihoods of the safety experts hired to oversee the project management depend on satisfying this "customer." Boards and panels that were originally set up as independent safety reviews and alternative reporting channels between levels, over time, were effectively taken over by the Project Office.

The CAIB report recommended the establishment of an Independent Technical Authority, but there needs to be more than one type and level of independent authority in an organization. For example, there should be an independent technical authority within the program but independent from the Program Manager and his/her concerns with budget and schedule. There also needs to be an independent technical authority outside the programs to provide organization-wide oversight and maintain standards.

Independent technical authority and review is also needed outside the projects and programs. For example, authority for tailoring or relaxing of safety standards should not rest with the project manager or even the program. The amount and type of safety applied on a program should be a decision that is made outside of the project, in a company at the corporate level and in a government agency like NASA at the headquarters level.

There also needs to be an external safety review process. The Navy, for example, achieves this review partly through a project-independent board called the Weapons System Explosives Safety Review Board (WSESRB) and an affiliated Software Systems Safety Technical Review Board (SSSTRB). WSESRB and SSSTRB assure the incorporation of explosives safety criteria in all weapon systems by reviews conducted throughout all the system's life cycle phases. Similarly, a Navy Safety Study Group is responsible for the study and evaluation of all Navy nuclear weapon systems. An important feature of these groups is that they are separate from the programs and thus allow an independent evaluation and certification of safety.

Safety Oversight and Communication: Insight versus oversight. Increased reliance on contracting necessitates more effective communication and more extensive safety oversight processes, not less.

In military procurement programs, oversight and communication is enhanced through the use of safety working groups. In establishing any type of oversight process, two extremes must be avoided: "getting in to bed" with the project and losing objectivity or backing off too far and losing insight. Working groups are an effective way of avoiding these extremes. They assure comprehensive and unified planning and action while allowing for independent review and reporting channels. Working groups usually operate at different levels of the organization.

As an example, the Navy Aegis system development was very large and included a System Safety Working Group at the top level chaired by the Navy Principal for Safety with permanent members being the prime contractor's system safety lead and representatives from various Navy offices.

Contractor representatives attended meetings as required. Members of the group were responsible for coordinating safety efforts within their respective organizations, for reporting the status of outstanding safety issues to the group, and for providing information to the WSESRB. Working groups also functioned at lower levels, providing the necessary coordination and communication for that level and to the levels above and below.

A surprisingly large percentage of the reports on recent aerospace accidents have implicated improper transitioning from an oversight to insight process. This transition implies the use of different levels of feedback control and a change from prescription management control to management by objectives, where the objectives are interpreted and satisfied according to the local context. For these ccidents, the change in management role from oversight to insight seems to have been implemented simply as a reduction in personnel and budgets without assuring that anyone was responsible for specific critical tasks.

As an example, the Mars Climate Orbiter accident report says NASA management of out-of-house missions was changed from oversight to insight—with far fewer resources devoted to contract monitoring. In Mars Polar Lander, there was essentially no JPL line management involvement or visibility into the software development and minimal involvement by JPL technical experts. Similarly, the MCO report suggests that authority and accountability were a significant issue in the accident and that roles and responsibilities were not clearly allocated. There was virtually no JPL oversight of Lockheed-Martin Astronautics subsystem development.

NASA is not the only group with this problem. The Air Force transition from oversight to insight was implicated in the April 30, 1999 loss of a Milstar-3 satellite being launched by a Titan IV/Centaur, as described in Chapter 8. The Air Force Space and Missile Center Launch Directorate and the 3rd Space Launch Squadron were transitioning from a task oversight to a process insight role. That transition had not been managed by a detailed plan and responsibilities, and how to perform them under the insight concept were not well defined.

13.0.7 Organizational Subsystems and Social Interaction Processes

As noted earlier, organizational subsystems include various communications system, information systems, reward and reinforcement systems, selection and retention systems, learning and feedback systems, career development systems, complaint and conflict resolution systems, and other such sub-systems. Each of these subsystems is staffed by technical experts who have to simultaneously fulfill three major functions: to deliver services to the programs/line operations, to monitor the programs/line operations to maintain standards (sometimes including legal compliance), and to help facilitate organizational transformation and change. In the context of system safety, each subsystem has an interdependent and contributing role. Furthermore, these roles interact with various social interaction processes, including leadership, teamwork, negotiations, problem solving, decision-making, partnership, entrepreneurship, and other such interaction processes. Vast literatures exist with respect to each dimension of this interaction process. Only two are covered here: (1) leadership and professional interactions and (2) safety information systems.

Leadership and Professional Interactions: Professional communication is critical for safety. In an interview shortly after he became Center Director at KSC, Jim Kennedy suggested that the most important cultural issue the Shuttle program faces is establishing a feeling of openness and honesty with all employees where everybody's voice is valued. Statements during the Columbia accident investigation and anonymous messages posted on the NASA Watch web site document a lack of trust of NASA employees to speak up. At the same time, a critical observation in the CAIB report focused on the managers' claims that they did not hear the engineers' concerns. The report concluded that this was due in part to the managers not asking or listening. Managers created barriers against dissenting opinions by stating preconceived conclusions based on subjective knowledge and experience rather than on solid data. In the extreme, they listened to those who told them what they wanted to hear. Just one indication of the atmosphere existing at that time were statements in the 1995 Kraft report that dismissed concerns about Shuttle safety by labeling those who made them as being partners in an unneeded "safety shield" conspiracy.

Changing such interaction patterns is not easy. Management style can be addressed through training, mentoring, and proper selection of people to fill management positions, but trust will take a while to regain. Carroll participated in culture change activities at the Millstone Nuclear Power Plant in 1996 due to a NRC review concluding there was an unhealthy work environment, which did not tolerate dissenting views and stifled questioning attitudes among employees [**?**]. The problems at Millstone are surprisingly similar to those at NASA and the necessary changes were the same: Employees needed to feel psychologically safe about reporting concerns and to believe that managers could be trusted to hear their concerns and to take appropriate action while managers had to believe that employees were worth listening to and worthy of respect. Through extensive new training programs and coaching, individual managers experienced personal transformations in shifting their assumptions and mental models and in learning new skills, including sensitivity to their own and others' emotions and perceptions. Managers learned to respond differently to employees who were afraid of reprisals for speaking up and those who simply lacked confidence that management would take effective action.

There is a growing body of literature on leadership that points to the need for more distributed models of leadership appropriate to the growing importance of network-based organizational structures. One intervention technique that is particularly effective in this respect is to have leaders serve as teachers [**?**]. Such activities pair leaders with expert trainers to help manage group dynamics, but the training itself is delivered by the program leaders. The Ford Motor Company used this approach as part of what they termed their Business Leadership Initiative (BLI) and have since extended it as part of their Safety Leadership Initiative (SLI). They found that employees pay more attention to a message delivered by their boss than by a trainer or safety official. Also, by learning to teach the materials, supervisors and managers are more likely to absorb and practice the key principles.

Safety Information Systems: Creating and sustaining a successful safety information system requires a culture that values the sharing of knowledge learned from experience. In lieu of a comprehensive information system, past success and unrealistic risk assessment are often used as the basis for decision-making.

Common problems in safety information systems are that necessary data is not collected and what is collected is often filtered and inaccurate; methods are lacking for the analysis and summarization of causal data; and information is not provided to decision makers in a way that is meaningful and useful to them.

As a consequence, learning from previous experience is delayed and fragmentary and use of the information in decision-making is limited. For example, hazard tracking and safety information systems are important sources for identifying the metrics and data to collect to use as leading indicators of potential safety problems and as feedback on the hazard analysis process. When numerical risk assessment techniques are used, operational experience can provide insight into the accuracy of the models and probabilities used. In various studies of the DC-10 by McDonnell Douglas, for example, the chance of engine power loss with resulting slat damage during takeoff was estimated to be less than one in a billion flights. However, this highly improbable event occurred four times in the DC-10s in the first few years of operation without raising alarm bells before it led to an accident and changes were made. Even one event should have warned someone that the models used might be incorrect.

Aerospace (and other) accidents have often involved unused reporting systems. In a Titan/Centaur/Milstar loss and in the Mars Climate Orbiter (MCO) accident, for example, there was evidence that a problem existed before the losses occurred, but there was no communication channel established for getting the information to those who could understand it and to those making decisions or, alternatively, the problem-reporting channel was ineffective in some way or was simply unused.

The MCO accident report states that project leadership did not instill the necessary sense of authority and accountability in workers that would have spurred them to broadcast problems they detected so that those problems might be "articulated, interpreted, and elevated to the highest appropriate level, until resolved." The report also states that "Institutional management must be accountable for ensuring that concerns raised in their own area of responsibility are pursued, adequately addressed, and closed out." The MCO report concludes that lack of discipline in reporting problems and insufficient follow-up was at the heart of the mission's navigation mishap. E-mail was used to solve problems rather than the official problem tracking system.

A critical deficiency in Mars Climate Orbiter project management was the lack of discipline in reporting problems and insufficient follow-up. The primary, structured problem-reporting procedure used by the Jet Propulsion Laboratory—the Incident, Surprise, Anomaly process—was not embraced by the whole team. The key issue here is not that the formal tracking system was bypassed, but understanding why this took place. What are the complications or risks for individuals in using the formal system? What makes the informal e-mail system preferable?

In the Titan/Centaur/Milstar loss, voice mail and e-mail were also used instead of a formal anomaly reporting and tracking system. The report states that there was confusion and uncertainty as to how the roll rate anomalies detected before flight (and eventually leading to loss of the satellite) should be reported, analyzed, documented and tracked. In all these accidents, the existing formal anomaly reporting system was bypassed and informal email and voice mail was substituted. The problem is clear but not the cause, which was not included in the reports and perhaps not investigated. When a structured process exists and is not used, there is usually a reason. Some possible explanations may be that the system is difficult or unwieldy to use or it involves too much overhead. There may also be issues of fear and blame that might be associated with logging certain kinds of entries in such as system. It may well be that such systems are not changing as new technology changes the way engineers work.

13.0.8 Capability and Motivation

[etc.]

Chapter 14

Where To From Here?

Bibliography

[1] D. Ackermann and M. J. Tauber, editors. *Mental Models and Human-Computer Interaction*. North-Holland, Amsterdam, 1990.

[2] Russell L. Ackoff. Towards a system of systems concepts. *Management Science*, 17(11):661–671, July 1971.

[3] Air Force Space Division. *System Safety Handbook for the Acquisition Manager*. SDP 127-1, January 12, 1987.

[4] James G. Andrus. Aircraft Accident Investigation Board Report: U.S. Army UH-60 Black Hawk Helicopters 87-26000 and 88-26060. Department of Defense, July 13, 1994.

[5] W.R. Ashby. *An Introduction to Cybernetics*. London: Chapman and Hall, 1956.

[6] W.R. Ashby. Principles of the self-organizing systems. in H. Von Foerster and G.W. Zopf (eds.) *Principles of Self-Organization*, Pergamon, 1962.

[7] Robert U. Ayres and Pradeep K. Rohatgi. Bhopal: Lessons for technological decision-makers. *Technology in Society*, 9:19–45, 1987.

[8] Edward Bachelder and Nancy Leveson. Describing and probing complex system behavior: A graphical approach. *Aviation Safety Conference*, Society of Automotive Engineers, Seattle, September 2001.

[9] Ludwig Benner Jr. Accident investigations: Multilinear events sequencing methods. *Journal of Safety Research*, 7(2):67–73, June 1975.

[10] Ludwig Bertalanffy. General Systems Theory: Foundations, Development, and Applications. G. Braziller, New York, 1969.

[11] M. Beveridge and E. Parkins. Visual representation in analogical program solving. *Memory and Cognition*, v. 15, 1987.

[12] William Bogard. *The Bhopal Tragedy*. Westview Press, Boulder, Colo., 1989.

[13] Richard C. Booten Jr. and Simon Ramo. The development of systems engineering. *IEEE Transactions on Aerospace and Electronic Systems*, AES-20(4):306–309, July 1984.

[14] D.R. Branscome (Chairman). WIRE Mishap Investigation Board Report. NASA, June 8, 1999.

[15] B. Brehmer. Dynamic Decision Making: Human Control of Complex Systems. *Acta Psychologica*, Vol. 81, pp. 211–241, 1992.

[16] Bureau of Air Safety Investigation. Advanced Technology Aircraft Safety Survey Report. Department of Transport and Regional Development, Australia, June 1996.

[17] M.A. Buttigieg and P.M. Sanderson. Emergent features in visual display design for two types of failure detection tasks. *Human Factors*, 33, 1991.

[18] Peter Checkland. *Systems Thinking, Systems Practice.* John Wiley & Sons, New York, 1981.

[19] Agnees Chisti. *Dateline Bhopal.* Concept Publishing Company, New Delhi, India, 1986.

[20] R.C. Conant and W.R. Ashby. Every good regulator of a system must be a model of that system. *International Journal of System Science*, 1, pp. 89–97, 1970.

[21] Richard I. Cook. Verite, Abstraction, and Ordinateur Systems in the Evolution of Complex Process Control. *3rd Annual Symposium on Human Interaction with Complex Systems (HICS '96)*, Dayton Ohio, August 1996.

[22] Council for Science and Society. *The Acceptability of Risks (The Logic and Social Dynamics of Fair Decisions and Effective Controls).* Barry Rose Publishers Ltd., 1977.

[23] B. Curtis, H. Krasner and N. Iscoe. A field study of the software design process for large systems. *Communications of the ACM*, 31(2): 1268–1287, 1988.

[24] DeKleer J, and J.S. Brown. Assumptions and ambiguities in mechanistic mental models. In D. Gentner and A.L. Stevens (eds.), *Mental Models*, Lawrence Erlbaum, 1983.

[25] Department of Employment. *The Flixborough Disaster: Report of the Court of Inquiry.* Her Majesty's Stationery Office, London, 1975.

[26] Ulli Diemer. Contamination: The Poisonous Legacy of Ontario's Environment Cutbacks. *Canada Dimension Magazine*, July–August, 2000.

[27] D. Dorner. On the difficulties people have in dealing with complexity. In Jens Rasmussen, Keith Duncan, and Jacques Leplat, editors, *New Technology and Human Error*, pages 97-109, John Wiley & Sons, New York, 1987.

[28] Nicolas Dulac. *Empirical Evaluation of Design Principles for Increasing Reviewability of Formal Requirements Specifications through Visualization.* MIT Master's Thesis, August 2003.

[29] Nicolas Dulac, Thomas Viguier, Nancy Leveson, and Margaret-Anne Storey. On the Use of Visualization in Formal Requirements Specification. *International Conference on Requirements Engineering*, Essen Germany, September 2002.

[30] K.D. Duncan. Reflections on fault diagnostic expertise. In Jens Rasmussen, Keith Duncan, and Jacques Leplat, editors, *New Technology and Human Error*, pages 261-269, John Wiley & Sons, New York, 1987.

[31] Paul Eddy, Elaine Potter, and Bruce Page, *Destination Disaster*. Quadrangle/N.Y. Times Book Co., 1976

[32] M. Edwards. The design of an accident investigation procedure. *Applied Ergonomics*, 12:111–115, 1981.

[33] W. Edwards. Dynamic decision theory and probabilistic information processing. *Human Factors*, 4, pp. 59–73, 1962.

[34] E.E. Euler, S.D. Jolly, and H.H. Curtis. The Failures of the Mars Climate Orbiter and Mars Polar Lander: A Perspective from the People Involved. *Guidance and Control*, American Astronautical Society, paper AAS 01-074, 2001.

[35] B. Fischoff, P. Slovic, and S. Lichtenstein. Fault trees: Sensitivity of estimated failure probabilities to problem representation. *Journal of Experimental Psychology: Human Perception and Performance*, vol. 4, 1978.

[36] F.R. Frola and C.O. Miller. System Safety in Aircraft Acquisition. Logistics Management Institute, Washington D.C., January 1984.

[37] Y. Fujita. What Shapes Operator Performance? JAERI Human Factors Meeting, Tokyo, November 1991.

[38] Harold Gehman (Chair). Columbia Accident Investigation Report. August 2003.

[39] R. Glaser and M. T. H. Chi. Overview. In R. Glaser, M. T. H. Chi, and M. J. Farr, editors, *The Nature of Expertise*. Erlbaum, Hillsdale, New Jersey, 1988.

[40] Sallie E. Gordon and Richard T. Gill. Cognitive Task Analysis. in Caroline E. Zsambok and Gary Klein (eds.), Naturalistic Decision Making, Lawrence Erlbaum Associates, 1997.

[41] William Haddon, Jr., The Prevention of Accidents. in Duncan W. Clark and Brian MacMahon (Eds.), *Preventive Medicine*, Little Brown and Company, 1967,

[42] Willie Hammer. *Product Safety Management and Engineering.* Prentice-Hall, 1980.

[43] Helicopter Accident Analysis Team. Final Report. NASA 1998.

[44] G. Harman. Logic, reasoning, and logic form. In *Language, Mind, and Brain*, T.W. Simon and R.J. Scholes (eds.), Lawrence Erlbaum Associates, 1982.

[45] Anthony A. Hoare. The Emperor's New Clothes. *Communications of the ACM*, Vol. 24, No. 2, January 1981, pp. 75–83.

[46] Andrew Hopkins. *Managing Major Hazards: The Lessons of the Moira Mine Disaster.* Allen & Unwin, Sidney, Australia, 1999.

[47] M.S. Jaffe. *Completeness, Robustness, and Safety of Real-Time Requirements Specification.* Ph.D. Dissertation, University of California, Irvine, 1988.

[48] Jaffe, M.S, Leveson, N.G., Heimdahl, M.P.E., and Melhart, B.E.. Software requirements analysis for real-time process-control systems. *IEEE Transations on Software Engineering*, SE-17(3):241–258, March 1991.

[49] William G. Johnson. *MORT Safety Assurance System*, New York: Marcel Dekker, 1980.

[50] JPL Special Review Board. Report on the Loss of the Mars Polar Lander and Deep Space 2 Missions. Nasa Jet Propulsion Laboratory, 22 March 2000.

[51] C.A. Kaplan and H.A. Simon. In search of insight. *Cognitive Psychology*, vol. 22, 1990.

[52] John G. Kemeny. Saving American Democracy: The Lessons of Three Mile Island. *Technology Review*, pp. 65–75, June-July 1980.

[53] Urban Kjellen. An evaluation of safety information systems at six medium-sized and large firms. *Journal of Occupational Accidents*, 3:273–288, 1982.

[54] Urban Kjellen. Deviations and the Feedback Control of Accidents. in Jens Rasmussen, Keith Duncan, and Jacques Leplat (Eds.), *New Technology and Human Error*, John Wiley & Sons, 1987.

[55] Gary A. Klein, Judith Orasano, R. Calderwood, and Caroline E. Zsambok (Editors). *Decision Making in Action: Models and Methods.* Ablex Publishers, 1993.

[56] Trevor A. Kletz. Human problems with computer control. *Plant/Operations Progress*, 1(4), October 1982.

[57] Trevor Kletz. Human problems with computer control. *Plant/Operations Progress*, 1(4), October 1982.

[58] John C. Knight and Nancy G. Leveson. An Experimental Evaluation of the Assumption of Independence in Multi-Version Programming. *IEEE Transactions on Software Engineering*, Vol. SE-12, No. 1, January 1986, pp. 96–109.

[59] K. Kotovsky, J.R. Hayes, and H.A. Simon. Why are some problems hard? Evidence from Tower of Hanoi. *Cognitive Psychology*, vol. 17, 1985.

[60] John Ladd. Bhopal: An essay on moral responsibility and civic virtue. Department of Philosophy, Brown University, Rhode Island, January 1987.

[61] Jerome Lederer. How far have we come? A look back at the leading edge of system safety eighteen years ago. *Hazard Prevention*, May/June 1986, pp. 8-10.

[62] Jacques Leplat. Accidents and incidents production: Methods of analysis. In Jens Rasmussen, Keith Duncan, and Jacques Leplat, editors, *New Technology and Human Error*, pp. 133–142, John Wiley & Sons, New York, 1987.

[63] Jacques Leplat. Occupational accident research and systems approach. In Jens Rasmussen, Keith Duncan, and Jacques Leplat, editors, *New Technology and Human Error*, pp. 181–191, John Wiley & Sons, New York, 1987.

[64] Jacques Leplat. Some observations on error analysis. In Jens Rasmussen, Keith Duncan, and Jacques Leplat, editors, *New Technology and Human Error*, pp. 311–316, John Wiley & Sons, New York, 1987.

[65] Nancy G. Leveson. High-Pressure Steam Engines and Computer Software. *IEEE Computer*, October 1994 (Keynote Address from IEEE/ACM International Conference on Software Engineering, 1992, Melbourne, Australia).

[66] Nancy G. Leveson. *Safeware: System Safety and Computers.* Addison Wesley, 1985.

[67] Nancy G. Leveson. Evaluating Accident Models using Real Aerospace Accidents. MIT SERL Technical Report, ESD-WP-2002-06, http://sunnyday.mit.edu/accidents/, June 2001.

[68] Nancy G. Leveson, Jon D. Reese, Sugi Koga, L. Denise Pinnel, and Sean D. Sandys. Analyzing Requirements Specifications for Mode Confusion Errors. *First International Workshop on Human Error and System Development*, Glasgow, 1997.

[69] N.G. Leveson, M. P.E. Heimdahl, H. Hildreth, and J.D. Reese. Requirements specification for process-control systems. *Trans. on Software Engineering*, SE-20(9), September 1994.

[70] D.C.R. Link (Chairman). Report of the Huygens Communications System Inquiry Board. NASA, December 2000.

[71] J.L. Lions. Ariane 501 Failure: Report by the Inquiry Board. European Space Agency, 19 July, 1996.

[72] D.A. Lucas. Mental models and new technology. In Jens Rasmussen, Keith Duncan, and Jacques Leplat, editors, *New Technology and Human Error*, pages 321-325. John Wiley & Sons, New York, 1987.

[73] Robyn R. Lutz. Analyzing Software Requirements Errors in Safety-Critical, Embedded Systems. *Proceedings of the International Conference on Software Requirements*, IEEE, January 1992, pp. 53–65.

[74] Robert E. Machol. The Titanic coincidence. *Interfaces*, 5(5):53–54, May 1975.

[75] Dale A. Mackall. Development and Flight Test Experiences with a Flight-Critical Digital Control System. NASA Technical Paper 2857, National Aeronautics and Space Administration, Dryden Flight Research Facility, November 1988.

[76] Karen Marais and Nancy G. Leveson. Archetypes for Organizational Safety. *Workshop on the Investigation and Reporting of Accidents*, September 2003.

[77] Mike W. Martin and Roland Schinzinger. *Ethics in Engineering.* McGraw-Hill Book Company, New York, 1989.

[78] Ralph F. Miles Jr. Introduction. In Ralph F. Miles Jr., editor, *Systems Concepts: Lectures on Contemporary Approaches to Systems*, pp. 1–12, John F. Wiley & Sons, New York, 1973.

[79] Gareth Morgan. *Images of Organizations.* Sage Publications, 1986.

[80] NASA/ESA Investigation Board. SOHO Mission Interruption. NASA, 31 August 1998.

[81] J.R. Newman. Extension of human capability through information processing and display systems. Technical Report SP-2560, System Development Corporation, 1966.

[82] D.A. Norman. *Things that Make us Smart.* Addison-Wesley Publishing Company, 1993.

[83] James Oberg. Why the Mars probe went off course. *IEEE Spectrum Magazine*, Vol. 36, No. 12, December 1999.

[84] Dennis R. O'Connor. *Report of the Walkerton Inquiry.* Ontario Ministry of the Attorney General, 2002.

[85] Elisabeth Pate-Cornell. Organizational aspects of engineering system safety: The case of offshore platforms. *Science*, 250:1210–1217, 30 November 1990.

[86] J.G. Pavlovich. Formal Report of the Investigation of the 30 April 1999 Titan IV B/Centaur TC-14/Milstar-3 (B32) Space Launch Mishap. U.S. Air Force, 1999.

[87] Charles Perrow. *Normal Accidents: Living with High-Risk Technology.* Basic Books, Inc., New York, 1984.

[88] Charles Perrow. The habit of courting disaster. *The Nation*, pp. 346–356, October 1986.

[89] William H. Pickering. Systems engineering at the Jet Propulsion Laboratory. In Ralph F. Miles Jr., editor, *Systems Concepts: Lectures on Contemporary Approaches to Systems*, pp. 125–150, John F. Wiley & Sons, New York, 1973.

[90] Joan L. Piper. *Chain of Events: The Government Cover-Up of the Black Hawk Incident and the Friendly Fire Death of Lt. Laura Piper.* Brasseys Inc., 2001.

[91] Simon Ramo. The systems approach. In Ralph F. Miles Jr., editor, *Systems Concepts: Lectures on Contemporary Approaches to Systems*, pps. 13–32, John F. Wiley & Sons, New York, 1973.

[92] J. Rasmussen. The Role of hierarchical knowledge representation in decision making and system management. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-15, no. 2, March/April 1985.

[93] J. Rasmussen. *Information Processing and Human–Machine Interaction: An Approach to Cognitive Engineering.* North Holland, 1986.

[94] J. Rasmussen. Mental models and the control of action in complex environments. In D. Ackermann and M.J. Tauber (eds.) *Mental Models and Human–Computer Interaction*, Elsevier (North-Holland), 1990, pp. 41–69.

[95] Jens Rasmussen. Human error and the problem of causality in analysis of accidents. In D.E. Broadbent, J. Reason, and A. Baddeley, editors, *Human Factors in Hazardous Situations*, pp. 1–12, Clarendon Press, Oxford, 1990.

[96] Jens Rasmussen. Risk Management in a Dynamic Society: A Modelling Problem. *Safety Science*, vol. 27, No. 2/3, Elsevier Science Ltd., 1997, pp. 183–213.

[97] Jens Rasmussen, Annelise Mark Pejtersen, and L.P. Goodstein. *Cognitive System Engineering*. John Wiley & Sons, 1994.

[98] Jens Rasmussen and Annelise Mark Pejtersen. Virtual ecology of work. In J. M. Flach, P. A. Hancock, K. Caird and K. J. Vicente, editors *An Ecological Approach to Human Machine Systems I: A Global Perspective*, Erlbaum, Hillsdale, New Jersey, 1995.

[99] Jens Rasmussen and Inge Svedung. *Proactive Risk Management in a Dynamic Society.* Swedish Rescue Services Agency, 2000.

[100] J. Reason. *Human Error.* Cambridge University Press, 1990.

[101] Rodriguez, M., Katahira, M., de Villepin, M., and Leveson, N.G. Identifying Mode Confusion Potential in Software Design, *Digital Aviation Systems Conference*, Philadelphia, October 2000.

[102] Nadine Sarter and David Woods. How in the world did I ever get into that mode?: Mode error and awareness in supervisory control. *Human Factors*, Vol. 37, No. 1, November 1995, pp. 5–19.

[103] Edgar Schein. *Organizational Culture and Leadership.* 2nd Edition, Sage Publications, 1986.

[104] Peter M. Senge. *The Fifth Discipline: The Art and Practice of Learning Organizations.* Doubleday Currency, New York, 1990.

[105] Nadine N. Sarter and David Woods. Strong, silent, and out-of-the-loop. CSEL Report 95-TR-01, Ohio State University, February 1995.

[106] Nadine Sarter, David D. Woods, and Charles E. Billings. Automation Surprises. in G. Salvendy (Ed.) *Handbook of Human Factors/Ergonomics*, 2nd Edition, Wiley, New York, in press.

[107] G.F. Smith. Representational effects on the solving of an unstructured decision problem. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-19, 1989, pp. 1083-1090.

[108] Scott A. Snook. *Friendly Fire.* Princeton University Press, 2000.

[109] Stephenson, A. Mars Climate Orbiter: Mishap Investigation Board Report. NASA, November 10, 1999.

[110] John D. Sterman. *Business Dynamics.* McGraw-Hill, 2000.

[111] Jouko Suokas. On the reliability and validity of safety analysis. Technical Report Publications 25, Technical Research Center of Finland, Espoo, Finland, September 1985.

[112] Donald H. Taylor. The role of human action in man–machine system errors. In Jens Rasmussen, Keith Duncan, and Jacques Leplat, editors, *New Technology and Human Error*, pp. 287–292, John Wiley & Sons, New York, 1987.

[113] U.S. Government Accounting Office, Office of Special Investigations. Operation Provide Comfort: Review of Air Force Investigation of Black Hawk Fratricide Incident (GAO/T-OSI-98-13). U.S. Government Printing Office, Washington, D.C. 1997.

[114] I. Vessey. Expertise in debugging computer programs: A process analysis. *Int. J. of Man–Machine Studies*, vol. 23, 1985.

[115] Kim J. Vicente. A Field Study of Operator Cognitive Monitoring at Pickering Nuclear Generating Station. Technical Report CEL 9504, Cognitive Engineering Laboratory, University of Toronto, 1995.

[116] Kim J. Vicente. *Cognitive Work Analysis: Toward Safe, Productive, and Healthy Computer-Based Work.* Erlbaum Associates, April 1999.

[117] K.J. Vicente, K. Christoffersen and A. Pereklit. Supporting operator problem solving through ecological interface design. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(4):529–545, 1995.

[118] K.J. Vicente and J. Rasmussen. Ecological interface design: Theoretical foundations. *IEEE Trans. on Systems, Man, and Cybernetics*, vol 22, No. 4, July/August 1992.

[119] Kenneth E.F. Watt. *The Titanic Effect.* Sinauer Associates, Inc., Stamford, Conn., 1974.

[120] Gerald Weinberg. *An Introduction to General Systems Thinking.* John Wiley & Sons, New York, 1975.

[121] E.L. Wiener. *Human Factors of Advanced Technology ("Glass Cockpit") Transport Aircraft.* NASA Contractor Report 177528, NASA Ames Research Center, June 1989.

[122] Earl L. Weiner and Renwick E. Curry. Flight-deck automation: Promises and problems. *Ergonomics*, 23(10):995–1011, 1980.

[123] Norbert Weiner. Cybernetics: or the Control and Communication in the Animal and the Machine. The MIT Press, 2nd Edition, 1965.

[124] Kathryn A. Weiss. *Building a Reusable Spacecraft Architecture using Component-Based System Engineering.* MIT Master's Thesis, August 2003.

[125] Kathryn A. Weiss, Elwin C. Ong, and Nancy G. Leveson. Reusable Software Architectures for Aerospace Systems. *Aircraft Engineering and Aerospace Technology*, Vol. 75, No. 5, 2003.

[126] Kathryn Weiss, Nancy Leveson, Kristina Lundqvist, Nida Farid, and Margaret Stringfellow. An Analysis of Causation in Aerospace Accidents. *Proceedings of the Digital Aviation Systems Conference*, Daytona, October 2001, pp. 137–147.

[127] David D. Woods. Some results on operator performance in emergency events. in D. Whitfield (ed.), *Ergonomic Problems in Process Operations*, Institute of Chemical Engineering Symposium, Ser. 90, 1984.

[128] D.D. Woods. Toward a theoretical base for representation design in the computer medium: Ecological perception and aiding human cognition. In J. M. Flach, P. A. Hancock, K. Caird and K. J. Vicente (Eds.), *An Ecological Approach to Human Machine Systems I: A Global Perspective*, Erlbaum, Hillsdale, New Jersey, 1995.

[129] David D. Woods. Lessons from beyond human error: Designing for resilience in the face of change and surprise. Design for Safety Workshop, NASA Ames Research Center, October 8-10, 2000.

[130] Thomas Young (Chairman). Mars Program Independent Assessment Team Report. NASA, March 2000.

[131] Mark Zimmerman, Nancy Leveson, and Kristina Lundqvist. Investigating the Readability of State-Based Formal Requirements Specification Languages. *Int. Conference on Software Engineering*, Orlando, May 2002.

[132] Caroline E. Zsambok and Gary Klein (Editors). *Naturalistic Decision Making.* Lawrence Erlbaum Associates, 1997.

Appendix A

Definitions

As people have been arguing about these definitions for decades, it is unlikely that everyone will agree with all (or perhaps even any) of the following definitions. They reflect, however, the use of these terms in this paper.

Accident An undesired and unplanned event that results in a loss (including loss of human life or injury, property damage, environmental pollution, etc.).

Safety Freedom from accidents (loss events).

Hazard A system state or set of conditions that, together with a particular set of worst-case environment conditions, will lead to an accident (loss).

Hazard Analysis The process of identifying hazards and their potential causal factors.

System Safety Engineering The system engineering processes used to prevent accidents by identifying and eliminating or controlling hazards. Note that hazards are not the same as failures; dealing with failures is usually the province of reliability engineering.

Hazard Level A function of the hazard *severity* (worst case damage that could result from the hazard given the environment in its most unfavorable state) and the *likelihood* (qualitative or quantitative) of its occurrence (Figure A.1).

Hazard Assessment The process involved in determining the hazard level.

Risk Factors Factors leading to an accident, including both hazards and the conditions or states of the environment associated with that hazard leading to an accident.

Risk Analysis The process of identifying risk factors and their potential causal factors.

Risk Level A function of the hazard level combined with (1) the likelihood of the hazard leading to an accident and (2) hazard exposure or duration.

Risk Assessment The process of determining the risk level (quantifying risk).

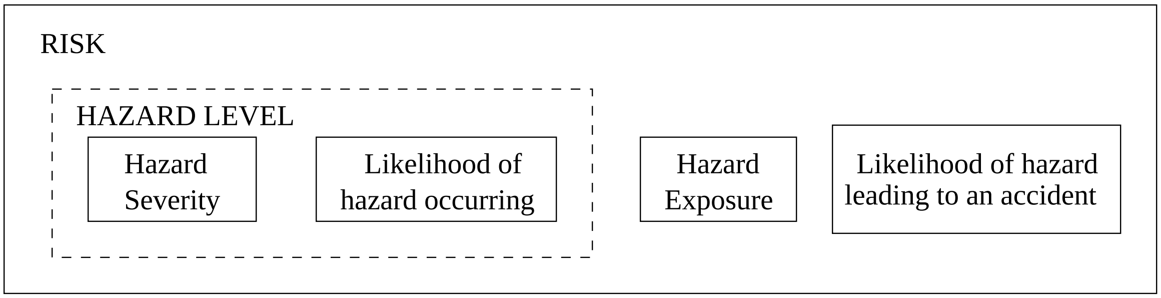

RISK
HAZARD LEVEL
Hazard Severity
Likelihood of hazard occurring
Hazard Exposure
Likelihood of hazard leading to an accident

Figure A.1: The Components of Risk.

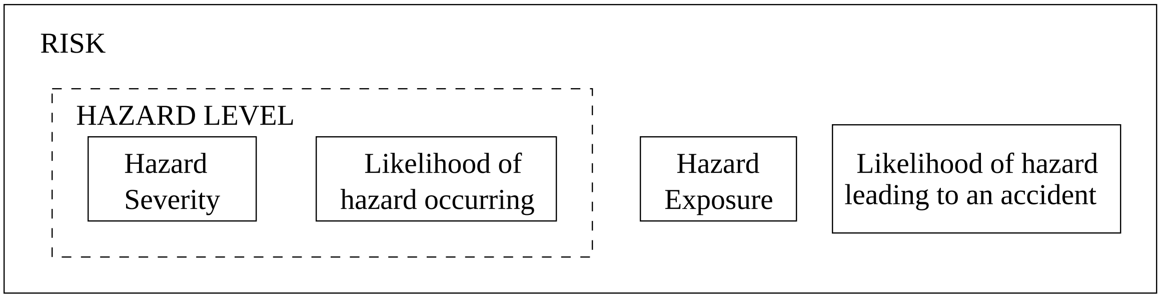

RISK
HAZARD LEVEL
Hazard Severity
Likelihood of hazard occurring
Hazard Exposure
Likelihood of hazard leading to an accident

Figure A.2: The Components of Risk.

Appendix B

The Role of Software in Spacecraft Accidents[1]

Software is playing an increasingly important role in aerospace systems. Is it also playing an increasing role in accidents and, if so, what type of role? In the process of a research project to evaluate accident models, I looked in detail at a variety of aerospace accidents that in some way involved software [67, 126] and was surprised at the similarity of the factors contributing to these losses.[2] To prevent accidents in the future, we need to attack these problems.

The spacecraft accidents investigated were the explosion of the Ariane 5 launcher on its maiden flight in 1996; the loss of the Mars Climate Orbiter in 1999; the destruction of the Mars Polar Lander sometime during the entry, deployment, and landing phase in the following year; the placing of a Milstar satellite in an incorrect and unusable orbit by the Titan IV B-32/Centaur launch in 1999; and the loss of contact with the SOHO spacecraft in 1998.

On the surface, the events and conditions involved in the accidents appear to be very different. A more careful, detailed analysis of the systemic factors, however, reveals striking similarities. These weaknesses not only contributed to the accident being investigated—for the Space Shuttle Challenger that might be a flawed design of the O-rings—but also can affect future accidents. For Challenger, the latter includes flawed decision making, poor problem reporting, lack of trend analysis, a "silent" or ineffective safety program, communication problems, etc.

The accidents are first briefly described for those unfamiliar with them, and then the common factors are identified and discussed. These factors are divided into three groups: (1) flaws in the safety culture, (2) management and organizational problems, and (3) technical deficiencies.

B.1 The Accidents

Ariane 501

On June 4, 1996, the maiden flight of the Ariane 5 launcher ended in failure. About 40 seconds after initiation of the flight sequence, at an altitude of 2700 m, the launcher veered off its flight path, broke up, and exploded. The accident report describes what they called the "primary cause" as the

[1] A version of this appendix is in press for publication in the AIAA Journal of Spacecraft and Rockets.

[2] The detailed analyses of these accidents and their causes using a hierarchical accident model can be found in [67, 126].

complete loss of guidance and attitude information 37 seconds after start of the main engine ignition sequence (30 seconds after liftoff) [71]. The loss of information was due to specification and design errors in the software of the inertial reference system. The software was reused from the Ariane 4 and included functions that were not needed for Ariane 5 but were left in for "commonality." In fact, these functions were useful but not required for the Ariane 4 either.

Mars Climate Orbiter (MCO)

The Mars Climate Orbiter (MCO) was launched December 11, 1998 atop a Delta II launch vehicle. Nine and a half months after launch, in September 1999, the spacecraft was to fire its main engine to achieve an elliptical orbit around Mars. It then was to skim through the Mars upper atmosphere for several weeks, in a technique called aerobraking, to move into a low circular orbit. On September 23, 1999, the MCO was lost when it entered the Martian atmosphere in a lower than expected trajectory. The investigation board identified what it called the "root" cause of the accident as the failure to use metric units in the coding of a ground software file used in the trajectory models [109]. Thruster performance data was instead in English units.

Mars Polar Lander (MPL)

Like MCO, Mars Polar Lander (MPL) was part of the Mars Surveyor program. It was launched January 3, 1999, using the same type of Delta II launch vehicle as MCO. Although the cause of the MPL loss is unknown, the most likely scenario is that the problem occurred during the entry, deployment, and landing (EDL) sequence when the three landing legs were to be deployed from their stowed condition to the landed position [50, 130]. Each leg was fitted with a Hall Effect magnetic sensor that generates a voltage when its leg contacts the surface of Mars. The descent engines were to be shut down by a command initiated by the flight software when touchdown was detected. The engine thrust must be terminated within 50 milliseconds after touchdown to avoid overturning the lander. The flight software was also required to protect against a premature touchdown signal or a failed sensor in any of the landing legs.

The touchdown sensors characteristically generate a false momentary signal at leg deployment. This behavior was understood and the flight software should have ignored it. The software requirements did not specifically describe these events, however, and consequently the software designers did not account for them. It is believed that the software interpreted the spurious signals generated at leg deployment as valid touchdown events. When the sensor data was enabled at an altitude of 40 meters, the software shut down the engines and the lander free fell to the surface, impacting at a velocity of 22 meters per second (50 miles an hour) and was destroyed.

Titan/Centaur/Milstar

On April 30, 1999, a Titan IV B-32/Centaur TC-14/Milstar-3 was launched from Cape Canaveral. The mission was to place the Milstar satellite in geosynchronous orbit. An incorrect roll rate filter constant zeroed the roll rate data, resulting in the loss of roll axis control and then yaw and pitch control. The loss of attitude control caused excessive firings of the reaction control system and subsequent hydrazine depletion. This erratic vehicle flight during the Centaur main engine burns in turn led to an orbit apogee and perigee much lower than desired, placing the Milstar satellite in an incorrect and unusable low elliptical final orbit instead of the intended geosynchronous orbit.

The accident investigation board concluded that failure of the Titan IV B-32 mission was due to a failed software development, testing, and quality assurance process for the Centaur upper stage [86]. That failed process did not detect the incorrect entry by a flight software engineer of a roll rate filter constant into the Inertial Navigation Unit software file.

The roll rate filter itself was included early in the design phase of the first Milstar spacecraft, but the spacecraft manufacturer later determined that filtering was not required at that frequency. A decision was made to leave the filter in place for the first and later Milstar flights for "consistency."

SOHO (SOlar Heliospheric Observatory)

SOHO was a joint effort between NASA and ESA to perform helioseismology and to monitor the solar atmosphere, corona, and wind. The spacecraft completed a successful two-year primary mission in May 1998 and then entered into its extended mission phase. After roughly two months of nominal activity, contact with SOHO was lost June 25, 1998. The loss was preceded by a routine calibration of the spacecraft's three roll gyroscopes and by a momentum management maneuver.

The flight operations team had modified the ground operations procedures as part of a ground systems reengineering effort to reduce operations costs and streamline operations, to minimize science downtime, and to conserve gyro life. Though some of the modifications were made at the request of the SOHO science team, they were not necessarily driven by any specific requirements changes. A series of errors in making the software changes along with errors in performing the calibration and momentum management maneuver and in recovering from the emergency safing mode led to the loss of telemetry [80]. Communication with the spacecraft was never restored.

B.2 Flaws in the Safety Culture

The *safety culture* is the general attitude and approach to safety reflected by those working in an industry. Safety culture flaws reflected in the accident reports include complacency and discounting the risks associated with software, confusing safety with reliability in software-intensive systems, assuming risk decreases over time, and ignoring warning signs.

B.2.1 Complacency and Discounting of Risks

Success is ironically one of the progenitors of accidents when it leads to overconfidence and cutting corners or making tradeoffs that increase risk. This phenomenon is not new, and it is extremely difficult to counter when it enters the engineering culture in an organization. Complacency is the root cause of most of the other accident factors described in this paper and was exhibited in all the accidents studied.

The Mars Climate Orbiter (MCO) report noted that because JPL's navigation of interplanetary spacecraft had worked well for 30 years, there was widespread perception that "orbiting Mars is routine" and inadequate attention was devoted to risk management. A similar culture apparently permeated the Mars Polar Lander (MPL) project.

In the SOHO loss, overconfidence and complacency, according to the accident report, led to inadequate testing and review of changes to ground-issued software commands to the spacecraft, a false sense of confidence in the team's ability to recover from a safe-hold mode (emergency sun reacquisition) from which a recovery sequence must be commanded and executed under ground operator control, the use of tight schedules and compressed timelines that eliminated any time to handle potential emergencies, inadequate contingency planning, responses to emergencies without taking the designed-in time to consider the options, etc. Protections built into the process, such as the review of critical decisions, were bypassed. After two previous SOHO spacecraft retreats to safe mode, the software and procedures were not reviewed because higher priority had been assigned to other tasks. The report concludes that the success in recovering from the previous safe mode entries led to overconfidence by the operations team in their ability to recover and a lack of appreciation of the risks involved in entering and recovering from the safing mode.

All the accidents involved systems built within an engineering culture that had unrealistic expectations about software and the use of computers. It is common for engineers to underestimate the complexity of most software and to overestimate the effectiveness of testing. The Ariane 5 accident report notes that software was assumed to be correct until it was shown to be faulty. The opposite assumption is more realistic.

In the Titan/Centaur accident, there apparently was no checking of the correctness of the software after the standard testing performed during development. For example, on the day of the launch, the attitude rates for the vehicle on the launch pad were not properly sensing the earth's rotation rate (the software was consistently reporting a zero roll rate) but no one had the responsibility to specifically monitor that rate data or to perform a check to see if the software attitude filters were operating correctly. In fact, there were no formal processes to check the validity of the filter constants or to monitor attitude rates once the flight tape was actually loaded into the Inertial Navigation Unit at the launch site. Potential hardware failures are usually checked up to launch time, but it may have been assumed that testing removed all software errors and no further checks were needed.

Complacency can also manifest itself in a general tendency of management and decision makers to discount unwanted evidence of risk. A *culture of denial* [46] arises in which any evidence of significant risk is dismissed.

A recommendation common to several of the spacecraft reports was to pay greater attention to risk identification and management. The investigators found that the project management teams appeared primarily focused on meeting mission cost and schedule objectives and did not adequately focus on mission risk. As an example, a report on the MPL loss concludes that the pressure of meeting the cost and schedule goals resulted in an environment of increasing risk in which too many corners were cut in applying proven engineering practices and in the checks and balances necessary for mission success.

While management may express their concern for safety and mission risks, true priorities are shown during resource allocation. MCO and MPL were developed under very tight "Faster, Better, Cheaper" budgets. The Titan program office had cut support for monitoring the software development and test process by 50% since 1994 and had greatly cut the number of engineers working launch operations. Although budget decisions are always difficult when resources are reduced—and budgets are almost always less than is optimal—the first things to be cut are often system safety, system engineering, quality assurance, and operations, which are assigned a low priority and assumed to be the least critical parts of the project.

B.2.2 Confusing Safety and Reliability in Software-Intensive Systems

Throughout the accident reports, there is an emphasis on failures as the cause of accidents. But accidents involving software are much more likely, in the author's experience, to be *system accidents* that result from dysfunctional interactions among components, not from individual component failure. Almost all the software accidents known to the author have resulted from the computer doing something wrong rather than the computer hardware or software failing to operate at all. In fact, each of the software or hardware components may have operated according to its specification (i.e., they did not fail), but the combined behavior of the components led to disastrous system behavior. All the accidents investigated for this paper displayed some aspects of system accidents.

System accidents are caused by interactive complexity and tight coupling [87]. Software allows us to build systems with a level of complexity and coupling that is beyond our ability to control; in fact, we are building systems where the interactions among the components (often controlled by software) cannot all be planned, understood, anticipated, or guarded against. This change is not solely the result of using digital components, but it is made possible because of the flexibility of software. Note that the use of redundancy only makes the problem worse—the added complexity introduced by redundancy has resulted in accidents that otherwise might not have occurred.

Engineering activities must be augmented to reflect the ways that software contributes to accidents. Almost all software-related aerospace accidents can be traced back to flaws in the requirements specification and not to coding errors—the software performed exactly as the designers intended (it did not "fail"), but the designed behavior was not safe from a system viewpoint. There is not only anecdotal but some hard data to support this hypothesis. Lutz examined 387 software errors uncovered during integration and system testing of the Voyager and Galileo spacecraft [73]. She concluded that the software errors identified as potentially hazardous to the system tended to be produced by different error mechanisms than non-safety-related software errors. She showed that for these two spacecraft, the safety-related software errors arose most commonly from (1) discrepancies between the documented requirements specifications and the requirements needed for correct functioning of the system and (2) misunderstandings about the software's interface with the rest of the system.

Software and digital systems require changes to some important aspects of engineering practice. Not only are failures not random (if the term "failure" makes any sense when applied to something like software that is pure design separated from the physical realization of that design), but the complexity of most software precludes examining all the ways it could "misbehave." And the failure modes (the way it misbehaves) can be very different than for physical devices. The JPL Mars Polar Lander accident report, like others, recommends using FMECA (Failure Modes and Effects Analysis) and FTA (Fault Tree Analysis) along with appropriate redundancy to eliminate failures. But these techniques were developed to cope with random wearout failures in hardware and are not very effective against design errors, the only type of error found in software.[5] The Ariane 5 accident report notes that according to the culture of the Ariane program, only random failures are addressed and they are primarily handled with redundancy. This approach obviously failed in the Ariane 5's first flight when both the primary and backup (redundant) Inertial Reference System computers shut themselves down—exactly as they were designed to do—as a result of the

[5] Although computer hardware can fail, software itself is pure design and thus all errors are design errors, even typos, which could be categorized as a type of inadvertent design error. One could argue that typos have a random aspect, but typos are usually caught in testing and have not contributed to a large number of spacecraft accidents. In any event, FTA and FMEA would not be very helpful in identifying potential software typos.

same unexpected input value.

To cope with software design errors, "diversity" has been suggested in the form of independent groups writing multiple versions of software with majority voting on the outputs. This approach is based on the assumption that such versions will fail in a statistically independent manner, but this assumption has been shown to be false in practice and by scientific experiments (see, for example, [58]). Common-cause (but usually different) logic errors tend to lead to incorrect results when the various versions attempt to handle the same unusual or difficult-to-handle inputs. In addition, such designs usually involve adding to system complexity, which can result in failures itself. A NASA study of an experimental aircraft with two versions of the control system found that all of the software problems occurring during flight testing resulted from errors in the redundancy management system and not in the control software itself, which worked perfectly [75].

The first step in handling system accidents is for engineers to recognize the need for change and to understand that system safety and component or even functional reliability are different qualities. For software especially, one does not imply the other. Although confusing reliability with safety is common in engineering (and particularly common in software engineering), it is perhaps most unfortunate with regard to software as it encourages spending much of the effort devoted to safety on activities that are likely to have little or no effect. In some cases, increasing component or system reliability actually decreases safety and vice versa.

B.2.3 Assuming Risk Decreases over Time

In the Milstar satellite loss, the Titan Program Office had decided that because the software was "mature, stable, and had not experienced problems in the past," they could use the limited resources available after the initial development effort to address hardware issues. In several of the accidents, quality and mission assurance as well as system engineering were also reduced or eliminated during operations because it was felt they were no longer needed or the resources were needed more elsewhere. In MCO, for example, the operations group did not even have a mission assurance manager.

During SOHO operations, there was a lack of analysis of prior emergency sun reacquisitions, inadequate staffing, no apparent mission assurance and quality assurance functions, inadequate attention paid to changes, etc. The SOHO Mission Management Plan required that the NASA Project Operations Director be responsible for programmatic matters, provide overall technical direction to the flight operations team, and interface with the ESA technical support director. The position had been descoped over time by NASA from a dedicated individual during launch and commissioning to one NASA individual spending less than 10% of his time tracking SOHO operations. ESA was to retain ownership of the spacecraft and to be responsible for its technical integrity and safety, but they were understaffed to perform this function in other than routine situations. It is very common to assume that risk is decreasing after an extended period of success and to let down one's guard.

In fact, risk usually increases over time, particularly in software-intensive systems, because caution wanes and safety margins are cut, because time increases the probability the unusual conditions will occur that trigger an accident, or because the system itself or its environment changes. In some cases, the introduction of an automated device may actually change the environment in ways not predicted during system design.

The Therac-25, a radiation therapy machine that massively overdosed five patients due to software flaws, operated safely thousands of times before the first accident [66]. As operators became more familiar with the Therac-25 operation, they started to type faster, which triggered a software error that had not surfaced previously. Similar changes in pilot behavior have been observed as they become more familiar with automation.

Software also tends to be frequently changed and "evolves" over time, The more changes that are made to software, the more the original design erodes and the more difficult it becomes to make changes without introducing errors. In addition, the assumptions and rationale behind the design decisions are commonly not documented and are easily violated when the software is changed. Changes to software appear to be easy and complacency can rear its ugly head again.

While it is indeed easy to change software, it is very difficult to change it correctly. Modifications to the SOHO command procedures were subjected to very little testing and review, perhaps because they were considered to be minor. The Mars Climate Orbiter software was changed to include a new thruster equation but a 4.45 correction factor (the difference between the metric and imperial units), buried in the original code, was not noticed when the new vendor-supplied equation was used to update the software [34].

To prevent accidents, all changes to software must be thoroughly tested and, in addition, analyzed for their impact on safety. Such change analysis will not be feasible unless special steps are taken during development to document the information needed. Incident and accident analysis, as for any system, are also important as well as performance monitoring and periodic operational process audits.

The environment in which the system and software are operating will change over time, partially as a result of the introduction of the automation or system itself. Basic assumptions made in the original hazard analysis process must have been recorded and then should be periodically evaluated to ensure they are not being violated in practice. For example, in order not to distract pilots during critical phases of flight, TCAS (an airborne collision avoidance system required on most commercial aircraft flying in U.S. airspace) includes the ability for the pilot to switch to a Traffic-Advisory-Only mode where traffic advisories are displayed but display of resolution advisories (escape maneuvers) is inhibited. It was assumed in the original TCAS system design and hazard analysis that this feature would be used only during final approach to parallel runways when two aircraft come close to each other and TCAS would call for an evasive maneuver. The actual use of this feature in practice would be an important assumption to check periodically to make sure it is not being used in other situations where it might lead to a hazard. But that requires that the assumption was recorded and not forgotten. It also assumes that a system hazard analysis was performed during the original system development.

B.2.4 Ignoring Warning Signs

Warning signs almost always occur before major accidents. In several of the accidents considered here, warning signs existed that the software was flawed but they went unheeded.

Engineers noticed the problems with the Titan/Centaur software after it was delivered to the launch site, but nobody seemed to take them seriously. A deficiency report on the difficulty in using the SOHO telemetry data interface had been submitted four years prior to the spacecraft loss, but never resolved. The problems experienced with the Mars Climate Orbiter (MCO) software during the early stages of the flight did not seem to raise any red flags. During the first four months of the MCO mission, the ground software angular momentum desaturation (AMD) files were not used in the orbit determination process because of multiple file format errors and incorrect spacecraft attitude data specifications. Four months were required to fix the files. Almost immediately (within a week) it became apparent that the files contained anomalous data that was indicating underestimation of the trajectory perturbations due to desaturation events. Despite all these hints that there were serious problems in the software and perhaps the development process, reliance was still placed on the supposedly fixed software without manual checks or alternative calculations.

A good argument can be made that it is unfair to judge too harshly those who have ignored warnings: Too many accident warnings that prove to be unfounded may desensitize those in decision-making positions and result in real warnings being ignored. Hindsight is always 20/20, and warning signs are always easier to identify after the accident has occurred. Nevertheless, people have a tendency to disregard events that do not lead to major accidents. Indications of potential problems almost always occur before major losses, and accidents could be prevented if these warning signs could be identified and heeded.

B.3 Management and Organizational Factors

The five accidents studied during this exercise, as well as most other major accidents, exhibited common organizational and managerial flaws, notably. a diffusion of responsibility and authority, limited communication channels, and poor information flow.

B.3.1 Diffusion of Responsibility and Authority

In most of the accidents, there appeared to be serious organizational and communication problems among the geographically dispersed partners. Responsibility was diffused without complete coverage and without complete understanding by anyone about what all the groups were doing. Roles were not clearly allocated. Both the Titan and Mars '98 programs were transitioning to process "insight" from process "oversight." Just as the MPL reports noted that "Faster, Better, Cheaper" was not defined adequately to ensure that it meant more than simply cutting budgets, this change in management role from oversight to insight seems to have been implemented simply as a reduction in personnel and budgets without assuring that anyone was responsible for specific critical tasks. One of the results of faster-better-cheaper was a reduction in workforce while maintaining an expectation for the same amount of work to be accomplished. In many of these accidents, the people were simply overworked—sometimes driven by their own dedication.

The MCO report concludes that project leadership did not instill the necessary sense of authority and accountability in workers that would have spurred them to broadcast problems they detected so that those problems might be "articulated, interpreted, and elevated to the highest appropriate level, until resolved." The Titan/Centaur accident also shows some of these same symptoms.

For SOHO, a transfer of management authority to the SOHO Project Scientist resident at Goddard Space Flight Center left no manager, either from NASA or ESA, as the clear champion of spacecraft health and safety. Instead, the transfer encouraged management decisions that maximized science return over spacecraft risk. In addition, the decision structure for real-time divergence from agreed-upon ground and spacecraft procedures was far from clear. The flight operations staff was apparently able to change procedures without proper review.

Inadequate transition from development to operations played a role in several of the accidents. Engineering management sometimes has a tendency to focus on development and to put less effort into planning the operational phase. The operations teams (in those accidents that involved operations) also seemed isolated from the developers. The MCO report notes this isolation and provides as an example that the operators did not know until long after launch that the spacecraft sent down tracking data that could have been compared with the ground data, which might have identified the software error while it could have been fixed. The operations crew for the Titan/Centaur also did not detect the obvious software problems, partly because of a lack of the knowledge required to detect them.

Most important, responsibility for safety does not seem to have been clearly defined outside of the quality assurance function on any of these programs. All the accident reports (except the Titan/Centaur) are surprisingly silent about their safety programs. One would think that the safety activities and why they had been ineffective would figure prominently in the reports.

Safety was originally identified as a separate responsibility by the Air Force during the ballistic missile programs of the 50s and 60s to solve exactly the problems seen in these accidents—to make sure that safety is given due consideration in decisions involving conflicting pressures and that safety issues are visible at all levels of decision making. An extensive system safety program was developed by NASA after the Apollo launch pad fire in 1967. But the Challenger accident report noted that the system safety program had become "silent" over time and through budget cuts. Has this perhaps happened again? Or are the system safety efforts just not handling software effectively?

One common mistake is to locate the safety efforts within the quality assurance function. Placing safety *only* under the assurance umbrella instead of treating it as a central engineering concern is not going to be effective, as has been continually demonstrated by these and other accidents. While safety is certainly one property (among many) that needs to be assured, it cannot be engineered into a design through after-the-fact assurance activities alone.

Having an effective safety program cannot prevent errors of judgment in balancing conflicting safety, schedule, and budget constraints, but it can at least make sure that decisions are informed and that safety is given due consideration. It also ensures that someone is focusing attention on what the system is not supposed to do, i.e., the hazards, and not just on what it is supposed to do. Both perspectives are necessary if safety is to be optimized.

B.3.2 Limited Communication Channels and Poor Information Flow

In the Titan/Centaur and Mars Climate Orbiter accidents, there was evidence that a problem existed before the loss occurred, but there was no communication channel established for getting the information to those who could understand it and to those making decisions or, alternatively, the problem-reporting channel was ineffective in some way or was simply unused.

All the accidents involved one engineering group not getting the information they needed from another engineering group. The MCO report cited deficiencies in communication between the project development team and the operations team. The MPL report noted inadequate peer communication and a breakdown in intergroup communication. The Titan/Centaur accident also involved critical information not getting to the right people. For example, tests right before launch detected the zero roll rate but there was no communication channel established for getting that information to those who could understand it. SOHO had similar communication problems between the operations team and technical experts. For example, when a significant change to procedures was implemented, an internal process was used and nobody outside the flight operations team was notified.

In addition, system engineering on several of the projects did not keep abreast of test results from all areas and communicate the findings to other areas of the development project: Communication is one of the most important functions in any large, geographically distributed engineering project and must be carefully planned and fostered.

Researchers have found that the second most important factor in the success of any safety program (after top management concern) is the quality of the hazard information system. Both collection of critical information as well as dissemination to the appropriate people for action is required. The MCO report concludes that lack of discipline in reporting problems and insufficient followup was at the heart of the mission's navigation mishap. In the Titan/Centaur loss, the use of voice mail and email implies there either was no formal anomaly reporting and tracking system or the formal reporting procedure was not known or used by the process participants for some reason. The report states that there was confusion and uncertainty as to how the roll rate anomalies should be reported, analyzed, documented and tracked because it was a "concern" and not a "deviation." There is no explanation of these terms.

In all the accidents, the existing formal anomaly reporting system was bypassed (in Ariane 5, there is no information about whether one existed) and informal email and voice mail was substituted. The problem is clear but not the cause, which was not included in the reports and perhaps not investigated. When a structured process exists and is not used, there is usually a reason. Some possible explanations may be that the system is difficult or unwieldy to use or it involves too much overhead. Such systems may not be changing as new technology changes the way engineers work.

There is no reason why reporting something within the problem-reporting system should be much more cumbersome than adding an additional recipient to email. Large projects have successfully implemented informal email processes for reporting anomalies and safety concerns or issues. New hazards and concerns will be identified throughout the development process and into operations, and there must be a simple and non-onerous way for software engineers and operational personnel to raise concerns and safety issues and get questions answered at any time.

B.4 Technical Deficiencies

These cultural and managerial flaws manifested themselves in the form of technical deficiencies: (1) inadequate system and software engineering, (2) inadequate review activities, (3) ineffective system safety engineering, (4) inadequate cognitive engineering, and (5) flaws in the test and simulation environments.

B.4.1 Inadequate System and Software Engineering

For any project as complex as those involved in these accidents, good system engineering is essential for success. In some of the accidents, system engineering resources were insufficient to meet the needs of the project. In others, the process followed was flawed, such as in the flowdown of system requirements to software requirements or in the coordination and communication among project partners and teams.

In the Titan project, there appeared to be nobody in charge of the entire process, i.e., nobody responsible for understanding, designing, documenting, controlling configuration, and ensuring proper execution of the process. The Centaur software process was developed early in the Titan program and many of the individuals who designed the original process were no longer involved in it due to corporate mergers and restructuring and the maturation and completion of the Titan/Centaur design and development. Much of the system and process history was lost with their departure and therefore nobody knew enough about the overall process to detect that it omitted any testing with the actual load tape or knew that the test facilities had the capability of running the type of test that could have caught the error.

Preventing *system* accidents falls into the province of *system* engineering—those building individual components have little control over events arising from dysfunctional interactions among components. As the systems we build become more complex (much of that complexity being made possible by the use of computers), system engineering will play an increasingly important role in the engineering effort. In turn, system engineering will need new modeling and analysis tools that can handle the complexity inherent in the systems we are building. Appropriate modeling methodologies will have to include software, hardware and human components of systems.

Given that software played a role in all the accidents, it is surprising the reports reflected so little investigation of the practices that led to the introduction of the software flaws and a dearth of recommendations to fix them. In some cases, software processes were declared in the accident reports to have been adequate when the evidence shows they were not.

These accidents all involved very common system and software engineering problems, including poor specification practices, unnecessary complexity and software functions, software reuse without appropriate safety analysis, and violation of basic safety engineering design practices in the digital components.

Poor or Missing Specifications

The vast majority of software-related accidents have been related to flawed requirements and misunderstanding about what the software should do. This experiential evidence points to a need for better specification review and analysis—the system and software specifications must be reviewable and easily understood by a wide range of engineering specialists.

All the reports refer to inadequate specification practices. The Ariane accident report mentions poor specification practices in several places and notes that the structure of the documentation obscured the ability to review the critical design decisions and their underlying rationale. Inadequate documentation of design rationale to allow effective review of design decisions is a very common problem in system and software specifications [3]. The Ariane report recommends that justification documents be given the same attention as code and that techniques for keeping code and its justifications consistent be improved.

The MPL report notes that the system-level requirements document did not specifically state the failure modes the requirement was protecting against (in this case possible transients) and speculates that the software designers or one of the reviewers might have discovered the missing requirement if they had been aware of the rationale underlying the requirements. The small part of the requirements specification shown in the accident report (which may very well be misleading) seems to avoid all mention of what the software should not do. In fact, standards and industry practices often forbid such negative requirements statements. The result is that software specifications often describe nominal behavior well but are very incomplete with respect to required software behavior under off-nominal conditions and rarely describe what the software is *not* supposed to do. Most safety-related requirements and design constraints are best described using such negative requirements or design constraints.

Not surprising, the interfaces were a source of problems. It seems likely from the evidence in several of the accidents that the interface documentation practices were flawed. The MPL report includes a recommendation that in the future "all hardware inputs to the software must be identified . . . The character of the inputs must be documented in a set of system-level requirements." This information is usually included in the standard interface specifications, and it is surprising that it was not.

There are differing accounts of what happened with respect to the MCO incorrect units problem. The official accident report seems to place blame on the programmers and recommends that the software development team be provided additional training in "the use and importance of following the Mission Operations Software Interface Specification (SIS)." Although it is not included in the official NASA Mars Climate Orbiter accident report, James Oberg in an IEEE Spectrum article on the accident [83] claims that JPL never specified the units to be used. It is common for specifications to be incomplete or not to be available until late in the development process. A different explanation for the error was provided by the developers [34]. According to them, the files were required to conform to a Mars Global Surveyor (MGS) heritage software interface specification. The equations used in the erroneous calculation were supplied by the vendor in English units.

> Although starting from MGS-heritage software, the coded MGS thruster equation had to be changed because of the different size RCS thruster that MCO employed (same vendor). As luck would have it, the 4.45 conversion factor, although correctly included in the MGS equation by the previous development team, was not immediately identifiable by inspection (being buried in the equation) or commented in the code in an obvious way that the MCO team recognized it. Thus, although the SIS required SI units, the new thruster equation was inserted in the place of the MGS equation—without the conversion factor[34].

This explanation raises questions about the other software specifications, including the requirements specification, which seemingly should include descriptions of the computations to be used. Either these did not exist or the software engineers did not refer to them when making the change. Formal acceptance testing apparently did not use the software interface specification because the test oracle (computed manually) used for comparison contained the same error as the output file [34] and thus testing did not detect the error. Similarly, informal interface testing by the Navigation team of ground software changes after launch did not test the correctness of the file formats (consistency with the interface specification) but concentrated on making sure the file could be moved across on the file server [34].

Complete and understandable specifications are not only necessary for development, but they are critical for operations and the handoff between developers, maintainers, and operators. In the Titan/Centaur accident, nobody other than the control dynamics engineers who designed the roll rate constants understood their use or the impact of filtering the roll rate to zero. When discrepancies were discovered right before the Titan/Centaur/Milstar launch, nobody understood them. The MCO operations staff also clearly had inadequate understanding of the automation and therefore were unable to monitor its operation effectively. The SOHO accident report mentions that no hard copy of the software command procedure set existed and the latest versions were stored electronically without adequate notification when the procedures were modified. The report also states that the missing software *enable* command (which led to the loss) had not been included in the software module due to a lack of system knowledge of the person who modified the procedure: he did not know that an automatic software function must be re-enabled each time Gyro A was despun. Such information, particularly about safety-critical features, obviously needs to be clearly and prominently described in the system specifications.

In some cases, for example the problems with the Huygens probe from the Cassini spacecraft, the designs or parts of designs are labeled as "proprietary" and cannot be reviewed by those who could provide the most important input [70]. Adequate system engineering is not possible when the system engineers do not have access to the complete design of the spacecraft.

Good specifications that include requirements tracing and design rationale are critical for complex systems, particularly those that are software-controlled. And they must be reviewable and reviewed in depth by domain experts.

Unnecessary Complexity and Software Functionality

One of the most basic concepts in engineering critical systems is to "keep it simple."

> The price of reliability is the pursuit of the utmost simplicity. It is a price which the very rich find most hard to pay [45].

The seemingly unlimited ability of software to implement desirable features often, as in the case of most of the accidents examined in this paper, usually pushes this basic principle into the background: *Creeping featurism* is a common problem in software-intensive systems.

The Ariane and Titan/Centaur accidents involved software functions that were not needed, but surprisingly the decision to put in or to keep (in the case of reuse) these unneeded features was not questioned in the accident reports. The software alignment function in the reused Ariane 4 software had no use in the different Ariane 5 design. The alignment function was designed to cope with the unlikely event of a hold in the Ariane 4 countdown: the countdown could be restarted and a short launch window could still be used. The feature had been used once (in 1989 in flight 33 of the Ariane 4). The Ariane 5 has a different preparation sequence and cannot use the feature at all. In addition, the alignment function computes meaningful results only before liftoff—during flight, it serves no purpose but the problem occurred while the function was operating after takeoff in the Ariane 5. The Mars Polar Lander accident also involved software that was executing when it was not necessary to execute, although in that case the function was required at a later time in the descent sequence.

The Titan/Centaur accident report explains that the software roll rate filter involved in the loss of the Milstar satellite was not needed but was kept in for consistency. The same justification is used to explain why the unnecessary software function leading to the loss of the Ariane 5 was retained from the Ariane 4 software. Neither report explains why consistency was assigned such high priority. While changing software that works can increase risk, executing unnecessary software functions is also risky.

For SOHO, there was no reason to introduce a new function into the module that eventually led to the loss. A software function already existed to perform the required maneuver and could have been used. There was also no need to despin Gyro A between gyro calibration and the momentum maneuvers. In all these projects, tradeoffs were obviously not considered adequately (considering the consequences), perhaps partially due to complacency about software risk.

The more features included in software and the greater the resulting complexity (both software complexity and system complexity), the harder and more expensive it is to test, to provide assurance through reviews and analysis, to maintain, and to reuse in the future. Engineers need to start making these hard decisions about functionality with a realistic appreciation of their effect on development cost and eventual system safety and system reliability.

Software Reuse without Appropriate Safety Analysis

Reuse and the use of commercial off-the-shelf software (COTS) is common practice today in embedded software development. The Ariane 5 software involved in the loss was reused from the Ariane 4. According to the MCO developers [34], the small forces software was reused from the Mars Global Surveyor project, with the substitution of a new thruster equation. Technical management accepted the "just like MGS" argument and did not focus on the details of the software.

It is widely believed that because software has executed safely in other applications, it will be safe in the new one. This misconception arises from confusion between software reliability and safety, as described earlier: most accidents involve software that is doing exactly what it was designed to do, but the designers misunderstood what behavior was required and would be safe, i.e., it reliably performs the wrong function.

The blackbox (externally visible) behavior of a component can only be determined to be safe by analyzing its effects on the system in which it will be operating, that is, by considering the specific operational context. The fact that software has been used safely in another environment provides *no* information about its safety in the current one. In fact, reused software is probably less safe because the original decisions about the required software behavior were made for a different system design and were based on different environmental assumptions. *Changing the environment in which the software operates makes all previous usage experience with the software irrelevant for determining safety.*

A reasonable conclusion to be drawn is not that software cannot be reused, but that a safety analysis of its operation in the new system context is mandatory: Testing alone is not adequate to accomplish this goal. For complex designs, the safety analysis required stretches the limits of current technology. For such analysis to be technically and financially feasible, reused software must contain only the features necessary to perform critical functions—another reason to avoid unnecessary functions.

COTS software is often constructed with as many features as possible to make it commercially useful in a variety of systems. Thus there is tension between using COTS versus being able to perform a safety analysis and have confidence in the safety of the system. This tension must be resolved in management decisions about specific project risk—ignoring the potential safety issues associated with COTS software can lead to accidents and potential losses that are greater than the additional cost would have been to design and build new components instead of buying them.

If software reuse and the use of COTS components are to result in acceptable risk, then system and software modeling and analysis techniques must be used to perform the necessary safety analyses. This process is not easy or cheap. Introducing computers does not preclude the need for good engineering practices nor the need for difficult tradeoff decisions, and it almost always involves higher costs despite the common myth that introducing automation, particularly digital automation, will save money.

Violation of Basic Safety Engineering Practices in the Digital Components

Although system safety engineering textbooks and standards include principles for safe design, software engineers are almost never taught them. As a result, software often does not incorporate basic safe design principles—for example, separating and isolating critical functions, eliminating unnecessary functionality, designing error-reporting messages such that they cannot be confused with critical data (as occurred in the Ariane 5 loss), and reasonableness checking of inputs and internal states.

Consider the Mars Polar Lander loss as an example. The JPL report on the accident states that the software designers did not include any mechanisms to protect against transient sensor signals nor did they think they had to test for transient conditions. Runtime reasonableness and other types of checks should be part of the design criteria used for any real-time software.

B.4.2 Inadequate Review Activities

General problems with the way quality and mission assurance are practiced were mentioned in several of the reports. QA often becomes an ineffective activity that is limited simply to checking boxes signifying the appropriate documents have been produced without verifying the quality of the contents. The Titan/Centaur accident report makes this point particularly strongly.

Review processes (outside of QA) are also described as flawed in the reports but few details are provided to understand the problems. The Ariane 5 report states that reviews including all major partners in the Ariane 5 program took place, but no information is provided about what types of reviews were held or why they were unsuccessful in detecting the problems. The MCO report recommends that NASA "conduct more rigorous, in-depth reviews of the contractor's and team's work," which it states were lacking on the MCO. The report also concludes that the operations team could have benefited from independent peer reviews to validate their navigation analysis technique and to provide independent oversight of the trajectory analysis. There is no mention of software quality assurance activities or the software review process in the MCO report.

In the MPL descent engine control software reviews, apparently nobody attending was familiar with the potential for spurious Hall Effect sensor signals. There have also been cases where concerns about proprietary software have prevented external reviewers familiar with the spacecraft from a systems viewpoint from reviewing the software (see, for example, the Cassini/Huygens communications link enquiry board report [70]).

The SOHO accident report states that the changes to the ground-generated commands were subjected to very limited review. The flight operations team placed high reliance on ESA and Matra Marconi Space representatives who were quite knowledgeable about the spacecraft design, but there were only two of them and neither was versed in the computer language used to define the commands. A simulation was performed on the new compressed SOHO timelines, but the analysis of a problem detected during simulation was still going on as the new procedures were being used.

The Ariane report says that the limitations of the inertial reference system software were not fully analyzed in reviews, and it was not realized that the test coverage was inadequate to expose such limitations. An assumption by the Ariane 5 developers that it was not possible to perform a complete system integration test made simulation and analysis even more important, including analysis of the assumptions underlying any simulation.

The Titan/Centaur report was the only one to mention the existence of an independent verification and validation review process by a group other than the developers. In that process, default values were used for the filter rate constants and the actual constants used in flight were never validated.

In general, software is difficult to review and the success of such an effort is greatly dependent on the quality of the specifications. However, identifying unsafe behavior, i.e., the things that the software should *not* do and concentrating on that behavior for at least part of the review process, helps to focus the review and to ensure that critical issues are adequately considered.

Such unsafe (or mission-critical) behavior should be identified in the system engineering process before software development begins. The design rationale and design features used to prevent the unsafe behavior should also have been documented and can be the focus of such a review. This presupposes, of course, a system safety process to provide the information, which does not appear to have existed for the projects that were involved in the accidents studied.

As mentioned earlier, almost all software-related accidents have involved incomplete requirements specification and unhandled or mishandled system states or conditions. The two identified Mars Polar Lander software errors, for example, involved incomplete handling of software states and are both examples of very common specification flaws and logic omissions often involved in accidents. Such errors are most likely to be found if spacecraft and subsystem experts participate actively in the reviews.

Software hazard analysis and requirements analysis techniques and tools exist to assist in finding these types of incompleteness. To make such a review feasible, the requirements should include only the externally visible (blackbox) behavior; all implementation-specific information should be put into a separate software design specification (which can be subjected to a later software design review by a different set of reviewers). The only information relevant for a software requirements review is the software behavior that is visible outside the computer. Specifying only blackbox behavior (in engineering terminology, the *transfer function* across the digital component) allows the reviewers to concentrate on the information of importance to them without being overwhelmed by internal design information that has no impact on externally observable behavior.

The language used to specify the software requirements is also critical to the success of such a review. The best way to find errors in the software requirements is to include a wide range of disciplines and expertise in the review process. They must be able to read and understand the specifications without extensive training and, ideally, the notation should not differ significantly from standard engineering notations. While formal and executable specification languages have tremendous potential for enhancing our ability to understand the implications of complex software behavior and to provide correct and complete requirements, most of the languages created by computer scientists require too much reviewer training to be practical. A high priority on readability and learnability has not been placed on the development of such languages.

B.4.3 Ineffective System Safety Engineering

All of the accident reports studied are surprisingly silent about the safety programs and system safety activities involved. Take SOHO for example. A hazard analysis surely would have shown that the roll rate and the status of gyros A and B were critical, and this information could have guided the design of feedback channels to the operators about their status. A rigorous system safety process also would have triggered special safety analysis when changes were made to the SOHO operational procedures involving safety-critical components. In addition, a strong system safety program would have ensured that high priority was given to the analysis of previous emergency sun reacquisitions, that greater controls were placed on safety-critical operational procedures, and that safety-related open operational reports, such as the one reporting the difficulty the SOHO operators were having in reviewing telemetry data, did not stay open for four years and instead were tracked and resolved in a timely manner.

There did appear to be a criticality analysis performed on many of these projects, albeit a flawed one. Several of the reports recommend reconsidering the definition they used of critical components, particularly for software. Unfortunately, not enough information is given about how the criticality analyses were performed (or in some cases if they were done at all) to determine why they were unsuccessful. Common practice throughout engineering, however, is to apply the same techniques and approaches that were used for electromechanical systems (e.g., FMEA and FMECA) to the new software-intensive systems. This approach will be limited because the contribution of software to accidents, as noted previously, is different than that of purely mechanical or electronic components. In particular, software does not fail in the sense assumed by these techniques.

Often hazard analyses simply omit software, and when included it is often treated superficially at best. The hazard analysis produced after the Mars Polar Lander loss is typical. The JPL report on the loss identifies the hazards for each phase of the entry, descent, and landing sequence, such as *Propellant line ruptures*, *Excessive horizontal velocity causes lander to tip over at touchdown*, and *Premature shutdown of the descent engines.* For software, however, only one hazard—*Flight software fails to execute properly*—is identified, and it is labeled as common to all phases.

The problem with such vacuous statements about software hazards is that they provide no useful information—they are equivalent to simply substituting the single statement *Hardware fails to operate properly* for all the other identified system hazards. What can engineers do with such general statements? Singling out the JPL engineers here is unfair because the same types of useless statements about software are common in the fault trees and other hazard analyses found in almost all organizations and industries. The common inclusion of a box in a fault tree or failure analysis that says simply *Software Failure* or *Software Error* can be worse than useless because it is untrue—all software misbehavior will not cause a system hazard in most cases—and it leads to nonsensical activities like using a general reliability figure for software (assuming one believes such a number can be produced) in quantitative fault tree analyses when such a figure does not reflect in any way the probability of the software exhibiting a particular hazardous behavior.

Software by itself is never dangerous—it is an abstraction without the ability to produce energy and thus to lead directly to a physical loss. Instead, it contributes to accidents through issuing (or not issuing) instructions to other components in the system. In the case of the identified probable factor in the MPL loss, the dangerous behavior was *Software prematurely shuts down the descent engines.* Such an identified unsafe behavior would be much more helpful during development in identifying ways to mitigate risk than the general statement *Software fails to execute properly.*

There are several instances of flawed risk tradeoff decisions associated with these accidents. For example, in the Ariane accident, there was a lack of effective analysis to determine which software variables should be protected during execution. Unfortunately, the accident reports describe flawed decisions, but not the process for arriving at them. Important information that is missing includes how the analyses and trade studies were performed and what additional information or additional analysis techniques could have allowed better decisions to be made.

Providing the information needed to make safety-related engineering decisions is the major contribution of system safety techniques to engineering. It has been estimated that 70-90% of the safety-related decisions in an engineering project are made during the early concept development stage [36]. When hazard analyses are not performed, are done only after the fact (for example, as a part of quality or mission assurance of a completed design), or are performed but the information is never integrated into the system design environment, they can have no effect on these decisions and the safety effort reduces to a cosmetic and perfunctory role.

The description of the MCO problem by the developers [34] says that the best chance to find and eliminate the problem existed at the early stages of development, but the team failed to recognize the importance of the small forces ground software and it was not given the same attention as the flight software.

The Titan/Centaur accident provides another example of what happens when such analysis is not done. The risk analysis, in that case, was not based on determining the steps critical to mission success but instead considered only the problems that had occurred in previous launches. Software constant generation (a critical factor in the loss) was considered to be low risk because there had been no previous problems with it. There is, however, a potentially enormous (perhaps unlimited) number of errors related to software and considering only those mistakes made previously, while certainly prudent, is not adequate.

Not only is such a *fly-fix-fly* approach inadequate for complex systems in general, particularly when a single loss is unacceptable, but considering only the specific events and conditions occurring in past accidents is not going to be effective when new technology is introduced into a system. Computers are, in fact, introduced in order to make radical changes in functionality and design. In addition, software is often used precisely because it is possible to make changes for each mission and throughout operations—the system being flown today is often not the same one that existed yesterday. Proper hazard analysis that examines all the ways the system components (including software) or their interaction can contribute to accidents needs to be performed and used in the original development and when making changes during operations.

At the same time, system-safety techniques, like other engineering techniques, need to be expanded to include software and the complex cognitive decision making and new roles played by human operators [66]. Existing approaches need to be applied, and new and better ones developed. Where appropriately modified system safety techniques have been used, they have been successful. If system hazard analysis is performed prior to software implementation (not just prior to testing, as is recommended in the MPL report), requirements can be analyzed for hazardous states and protection against potentially hazardous behavior designed into the software logic from the beginning.

The Mars Climate Orbiter accident report recommended that the NASA Mars Program institute a classic system safety engineering program, i.e.,

- Continually performing the system hazard analyses necessary to explicitly identify mission risks and communicating these risks to all segments of the project team and institutional management;
- Vigorously working to make tradeoff decisions that mitigate the risks in order to maximize the likelihood of mission success; and
- Regularly communicating the progress of the risk mitigation plans and tradeoffs to project, program, and institutional management.

The other spacecraft accident reports, in contrast, recommended applying classic reliability engineering approaches that are unlikely to be effective for system accidents or software-related causal factors.

One of the benefits of using system-safety engineering processes is simply that someone becomes responsible for ensuring that particular hazardous behaviors are eliminated if possible or their likelihood reduced and their effects mitigated in the design. Almost all attention during development is focused on what the system and software are supposed to do. A system safety engineer or software safety engineer is responsible for ensuring that adequate attention is also paid to what the system and software are *not* supposed to do and verifying that hazardous behavior will not occur. It is this unique focus that has made the difference in systems where safety engineering successfully identified problems that were not found by the other engineering processes.

B.4.4 Flaws in the Test and Simulation Environments

It is always dangerous to conclude that poor testing was the "cause" of an accident. After the fact, it is always easy to find a test case that would have uncovered a known error. It is usually difficult, however, to prove that the particular test case would have been selected beforehand, even if testing procedures were changed. By definition, the cause of an accident can always be stated as a failure to test for the condition that was determined, after the accident, to have led to the loss. However, in the accidents studied, there do seem to be omissions that reflect poor decisions related to testing, particularly with respect to the accuracy of the simulated operational environment.

A general principle in testing aerospace systems is to *fly what you test and test what you fly.* This principle was violated in all the spacecraft accidents, especially with respect to software. The software test and simulation processes must reflect the environment accurately. Although implementing this principle is often difficult or even impossible for spacecraft, no reasonable explanation was presented in the reports for some of the omissions and flaws in the testing for these systems. An example was the use of Ariane 4 trajectory data in the specifications and simulations of the Ariane 5 software even though the Ariane 5 trajectory was known to be different. Another example was not testing the Titan/Centaur software with the actual load tape prior to launch. Testing of SOHO operational procedures was primarily performed using a simulator, but the simulator had not been maintained with all the on-board software changes that had been implemented on the spacecraft, essentially making such testing useless.

For both the Ariane 5 and Mars '98 projects, a conclusion was reached during development that the components implicated in the accidents could not be tested and simulation was substituted. After the fact, it was determined that such testing was indeed possible and would have had the ability to detect the design flaws. The same occurred with the Titan/Centaur accident, where default and simulated values were used in system testing although the real roll rate filter constants could have been used. Like Ariane, the Titan/Centaur engineers incorrectly thought the rigid-body simulation of the vehicle would not exercise the filters sufficiently. Even the tests performed on the Titan/Centaur right before launch (because anomalies had been detected) used default values and thus were unsuccessful in detecting the error. After wiring errors were discovered in the MPL testing process, for undisclosed reasons the tests necessary to detect the software flaw were not rerun.

Not all problems in testing can be traced to the simulation environment, of course. There have been cases of spacecraft losses involving inadequate, inappropriate, and ill-suited testing. A basic problem is one of piecemeal testing and not testing at the system level for system-level effects and emergent behavior. The rush to get the ground software operational after a problem was discovered post-launch in MCO resulted in the testing program being abbreviated. At a November 10, 1999 press conference, Alfred Stephenson, the chief accident investgator, admitted, “Had we done end-to-end testing, we believe this error would have been caught.” But the rushed and inadequate preparations left no time to do it right. The problem lies not only in testing, but in relying on software that had not been adequately tested without additional manual or other checks to gain confidence. The same lack of system test also contributed to the WIRE (Wide-Field Infrared Explorer Mission) spacecraft where a contributing cause cited in the accident report was that no system level end-to-end test with live pyrotechnic devices in the as-flown configuration had been done [14].

A final very common problem in software testing is inadequate emphasis on off-nominal and stress testing.

Better system testing practices are needed for components containing software (almost everything these days), more accurate simulated environments need to be used in software testing, and the assumptions used in testing and simulations need to be carefully checked.

B.5 Inadequate Cognitive Engineering

Cognitive engineering, particularly that directed at the influence of software design on human error, is still in its early stages. Human factors experts have written extensively on the potential risks introduced by the automation capabilities of glass cockpit aircraft. Among those identified are: mode confusion and situational awareness difficulties; inadequate feedback to support effective monitoring and decision making; over reliance on automation; shifting workload by increasing it during periods of already high workload and decreasing it during periods of already low workload; being “clumsy” or difficult to use; being opaque or difficult to understand; and requiring excessive experience to gain proficiency in its use. Accidents, surveys and simulator studies have emphasized the problems pilots are having in understanding digital automation and have shown that pilots are surprisingly uninformed about how the automation works [16, 102]. Not all of this information seems to have affected engineering practice: After commercial aircraft accidents, it is more common to simply blame the pilot for the accident than to investigate the aspects of system design that may have led to the human error(s).

As more sophisticated automation has been introduced into spacecraft control and control of safety-critical functions is increasingly shared between humans and computers, the same problems found in high-tech aircraft are appearing. Neither the Mars Climate Orbiter nor the Titan mission operations personnel understood the system or software well enough to interpret the data they saw as indicating there was a problem in time to prevent the loss. Complexity in the automation combined with poor documentation and training procedures are contributing to these problems.

Problems abound in the design of the interfaces between humans and automation. The SOHO operations personnel had filed a report (which had not been resolved in four years) about the difficulty they were having interpreting the telemetry data using the interface they were given. In addition, several places in the SOHO accident report hint at the controllers not having the information they needed about the state of the gyros and the spacecraft in general to make appropriate decisions. The misdiagnosis of Gyro B as the bad one and its subsequent deactivation raises many important questions about the information provided to the operators that are not answered in the accident report.

Complexity in the automation combined with poor documentation and training procedures are contributing to the problems we are seeing. Sometimes the incorrect assumption is made that introducing computers lessens the need for in-depth knowledge by operational personnel but the opposite is true. Some of the spacecraft accidents where operators were implicated involved a failure to transfer skill and knowledge from those who operated spacecraft in prior missions, and they were therefore unable to detect the software deficiencies in time to save the mission.

Either the design of the automation we are building needs to be improved from a cognitive engineering viewpoint or new training methods are needed for those who must deal with the clumsy automation and confusing, error-prone interfaces we are designing.

B.6 Conclusions

Complacency and misunderstanding software and its risks were at the root of all these accidents. Software presents tremendous potential for increasing our engineering capabilities. At the same time, it introduces new causal factors for accidents and requires changes in the techniques used to prevent the old ones. We need to apply the same good engineering practices to software development that we apply to other engineering technologies while also understanding the differences and making the appropriate changes to handle them. There is no magic in software—it requires hard work and is difficult to do well, but the result is worth the effort.

Appendix C

A Brief Introduction to State Machines

Appendix D

SpecTRM-RL Description

SpecTRM (Specification Tools and Requirements Methodology) is a system engineering toolset that focuses on the early stages of system development, where the foundation is set for later implementation, operations, and maintenance activities. The SpecTRM toolset includes support for requirements development and management, hazard analysis, requirements tracing, recording of design rationale, and modeling and analysis of blackbox logic requirements. The formal modeling language, SpecTRM-RL (SpecTRM Requirements Language), is executable and therefore can be used in a simulation environment. In addition, the models are analyzable and tools have been developed to check for completeness, consistency, and robustness.

The design of SpecTRM-RL is greatly influenced by the desire to provide a combined specification and modeling language. System specifications (particularly requirements specifications) need to be reviewed and used by people with a large variety of backgrounds and expertise, most of whom are not computer scientists or trained in formal logic or discrete math and may not even be engineers. Therefore, it is not practical to use most formal modeling languages as specification languages. On the other hand, industrial projects rarely have the resources to provide a separate modeling effort for the specification, and the continual changes common to most software development projects will require frequent updates to ensure that the formal model is consistent with the current requirements and system design.

SpecTRM-RL was designed to satisfy both objectives: to be easily readable enough to serve as part of the official specification of the blackbox behavioral requirements and, at the same time, to have an underlying formal model that can be executed and subjected to mathematical analysis.

This underlying formal model based on a Mealy automaton, which we call the requirements state machine (RSM), is very low-level and not appropriate as a specification language for complex systems. Instead, SpecTRM-RL acts as the specification language (or visualization of the underlying model) that overlays the low-level model. As long as the mapping from SpecTRM-RL to the RSM is unambiguous and well-defined, formal analysis is possible on both the underlying RSM formal model as well as the higher-level SpecTRM-RL specification itself. Leveson and her students at MIT are experimenting with additional capabilities to augment the current SpecTRM-RL features, and these will be added eventually to the commercial language.

To assist in readability, we use graphical, tabular, symbolic, and structural notations where we have found each most appropriate for the type of information being specified. Decisions about how things are specified were based on the research literature on visualization, feedback from users of RSML (a previous version of the language) and SpecTRM-RL, our own attempts to build

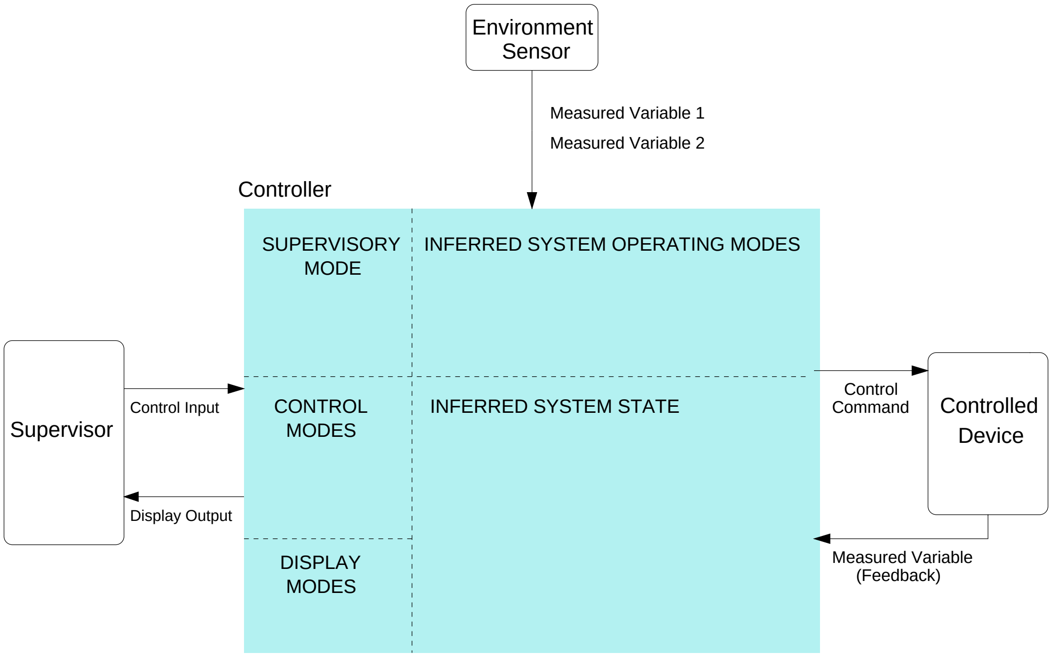


Figure D.1: The Parts of a SpecTRM-RL Graphical Model

specifications for real systems using the language, observation of the notations engineers use for specifying these properties, and formal experiments on readability of particular types of language features [131].

The SpecTRM-RL notation is driven by the intended use of the language to define a blackbox function from outputs to inputs. Some features are also the result of wanting to remain as close as possible to the way engineers draw and define control loops in order to enhance usability of the language. Other goals for the language design were to eliminate error-prone features, to make the language easy to learn and easy to read and review, and to encourage completeness in specifications.

The SpecTRM-RL language includes a graphical overview of the system structure along with specification of output messages, inputs, state variables, macros, and functions. The rest of this appendix describes each of these features.

Graphical Specification of the System Model : Figure D.1 shows the four main components of a SpecTRM-RL specification: (1) a specification of the supervisory modes of the controller being modeled, (2) a specification of its control modes (3) a model of the controlled process (or plant in control theory terminology) that includes the inferred operating modes and system state (these are inferred from the measured inputs), and (4) a specification of the inputs and outputs to the controller. The graphical notation mimics the typical engineering drawing of a control loop.

Every automated controller has at least two interfaces: one with the supervisor(s) that issues instructions to the automated controller (the supervisory interface) and one with each controlled system component (controlled system interface). The supervisory interface is shown to the left of the main controller model while the interface with the controlled component is shown to the right.

The supervisory interface consists of a model of the operator controls and a model of the displays or other means of communication by which the component relays information to the supervisor. Note that the interface models are simply the logical view that the controller has of the interfaces—the real state of the interface may be inconsistent with the assumed state due to various types of design flaws or failures. By separating the assumed interface from the real interface, it is possible to model and analyze the effects of various types of errors and failures (e.g., communication errors or display hardware failures). In addition, separating the physical design of the interface from the logical design (required content) will facilitate changes and allow parallel development of the software and the interface design. During development, mockups of the physical screen or interface design can be generated and tested using the output of the SpecTRM-RL simulator.

The bottom left quadrant of Figure D.1 provides information about the control modes of the controller itself. These are not internal states of the controller (which are not included in our specifications) but simply represent externally visible behavior about the controller's modes of operation (described further below).

The right half of the controller model represents inferred information about the operating modes and states of the controlled system (the plant in control theory terminology). The model for a simple plant like a thermostat might include only one or two variables while that for a more complex system, e.g., air traffic control, might contain a large number of variables and include operational modes and models of multiple subcomponents. In a hierarchical control system, the controlled process may itself be a controller of another process. For example, the flight management system may be controlled by a pilot and may issue commands to a flight control computer, which issues commands to an engine controller. Parts of a SpecTRM-RL model can be reused or changed to represent different members of a product family [125].

Figure D.2 shows the graphical part of a SpecTRM-RL specification of a simple altitude switch. The specification is based on an unpublished specification of an altitude switch by Steve Miller at Rockwell Collins. This switch turns on a Device of Interest (DOI) when the aircraft descends through a threshold altitude.

In SpecTRM-RL, state values in square boxes in the right side of the diagram represent inferred values used in the control of the computation of the blackbox I/O function. Such variables are necessarily discrete in value [1] and thus can be represented as a state variable with a finite number of possible values. In practice, such state variables almost always have only a few relevant values (e.g., altitude below a threshold, altitude at or above a threshold, cannot-be-determined, and unknown). Values for state variables in the plant model are required in SpecTRM-RL to include an *unknown* value. The meaning and purpose of the unknown state value are described below.

In the altitude switch example, defining the control algorithm requires using information about the aircraft altitude level with respect to a given threshold, the inferred status of the DOI, and the validity of the altimeter information being provided as well as the measured variables and various constants defined elsewhere in the specification.

The possible values for a state variable are shown with a line connecting the boxes. The line simply denotes that the values are disjoint, that is, the variable may assume only one value at a time. A small arrow pointing at a box denotes the default (startup) value for the state variable or mode. For example, the DOI-Status can have the values *On, Off, Unknown*, and *Fault-Detected.* The default value is *Unknown.*

The altitude switch has two control inputs (shown on arrows to the left of the Component

[1] If they are not discrete, then they are not used in the control of the function computation, but in the computation itself and can simply be represented in the specification by arithmetic expressions involving input variables.

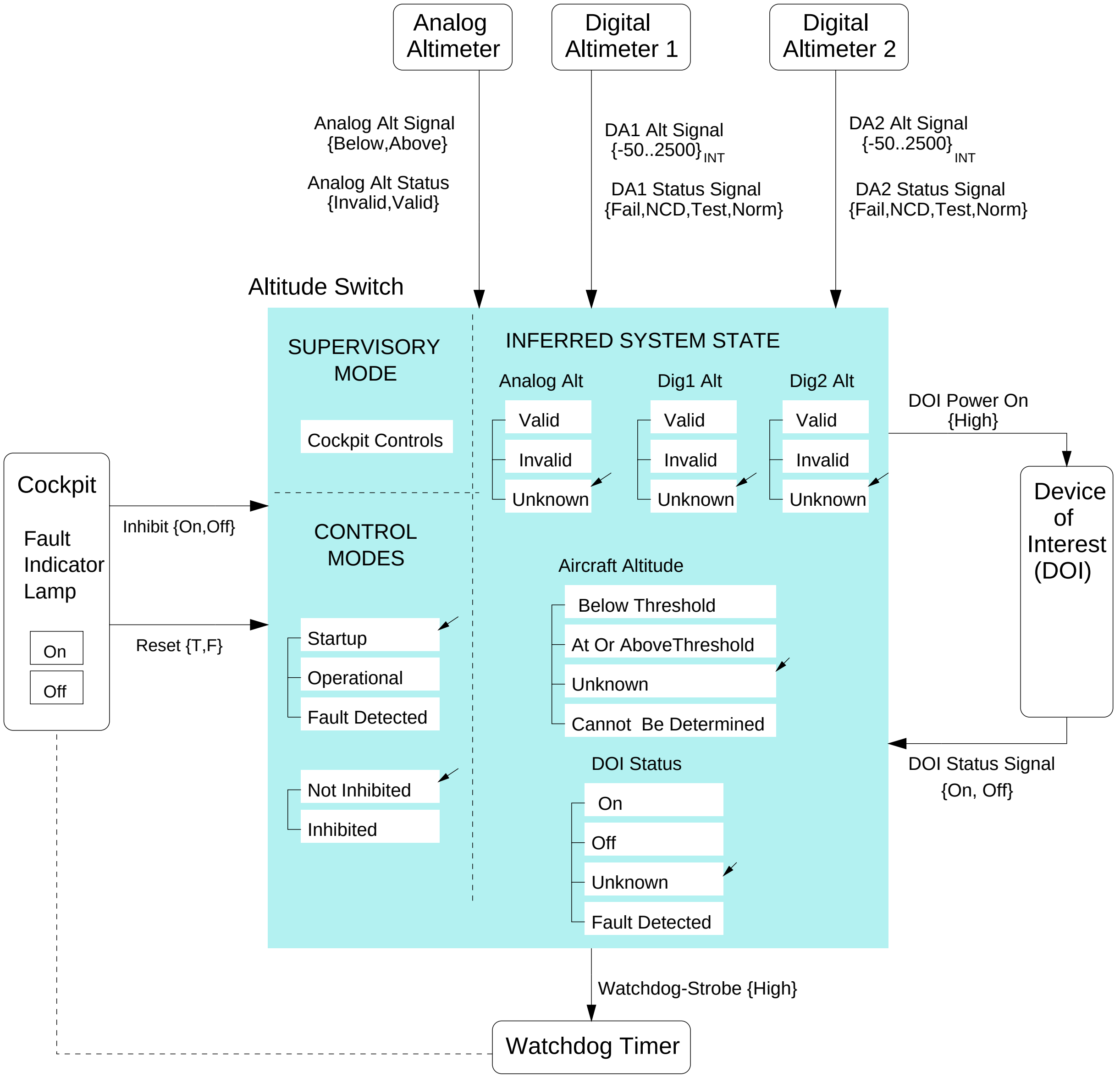


Figure D.2: Example of a Graphical Model

diagram): a reset signal that has the value true or false and an inhibit button that inhibits operation of the altitude switch. The inhibit button can either be in the *on* or *off* position. The only display in the altitude switch example is a fault indicator lamp that can also either be on or off, but its content is controlled through the watchdog timer and not directly by the altitude switch. There is only one supervisory mode—cockpit controlled—which is shown in the upper left quadrant of the component model.

Inputs representing the state of the plant (monitored or measured variables) are shown with arrows pointing to the controller. For the altitude switch, these variables provide (1) the current status (on or off) of the device of interest (DOI) that the altitude switch turns on and (2) inputs about the status and value of three altimeters on the aircraft (one analog and two digital) that provide information to the altitude switch about the current measured altitude of the aircraft as well as the status of that information (i.e., normal operation, test data, no computed data provided, or failed).

The output commands are denoted by outward pointing arrows. In the example, they include a signal to *power-on* the device (DOI) and a *strobe* to a watchdog timer so that proper action can be taken (by another system component) if the altitude switch fails. The outputs in this example are simple "high" signals on a wire or line to the device.

Note that the internal design of the altitude switch is not included in the model. The altitude switch operating modes are externally visible (and must be known for the pilot to understand its operation) and the aircraft model is used to describe the externally visible behavior of the altitude switch in terms of the process being controlled (and not in terms of its own internal data structures and algorithms). Thus the specification is blackbox.

Because of the simplicity of the altitude switch example, there are a few features of SpecTRM-RL that are not needed or are not well illustrated. Almost all of the missing features involve the ability to specify modes. Modes are abstractions on states and are not necessary for defining blackbox behavior. They are useful, however, in understanding or explaining the behavior of complex systems. While some formal specification languages use the term "mode" as a synonym for state (all modes are states and vice versa), SpecTRM-RL uses the more limited definition of mode common in engineering, i.e., as a state variable that plays a particular role in the state machine. In this usage, modes partition the state space into disjoint sets of states. For example, the state machine may be in normal operational mode or in a maintenance mode. Our definition was chosen to assist in reviewability of the specification by domain experts and in formal analysis of specifications for particular properties commonly involved in operator mode confusion [68]. SpecTRM-RL allows specifying several types of modes: supervisory modes, control modes, controlled-system operating modes, and display modes.

Supervisory modes are useful when a component may have multiple supervisors at any time. For example, a flight control computer in an aircraft may get inputs from the flight management computer and also directly from the pilot. Required behavior may differ depending on which supervisory mode is currently in effect. Mode-awareness errors related to confusion in coordination between multiple supervisors can be defined (and the potential for such errors theoretically identified from the models) in terms of these supervisory modes.

Control Modes control the behavior of the controller itself. Modern avionics systems may have dozens of modes. Control modes may be used in the interpretation of the component's interfaces or to describe the component's required process-control behavior. In the altitude switch, two types of control modes are useful in specifying the blackbox behavior: (1) whether the switch is in the *startup, operational*, and *internal-fault-detected* mode (the latter will result in the fault indicator light being lit in the cockpit and cessation of activity until the reset button is pushed) and (2) whether the operation of the altitude switch is partially inhibited or not. These two sets of modes cannot be combined in this case as they are not disjoint. In fact, in our original specification of the altitude switch, they were combined. We later found that error through the use of our completeness criteria.

A third type of mode, *controlled-system or plant operating modes*, can be used to specify sets of related behaviors of the controlled-system (plant) model. They are used to indicate its operational status. For example, it may be helpful to define the operational state of an aircraft in terms of it being in takeoff, climb, cruise, descent, or landing mode. Such operating modes are not needed to define the behavior of the altitude switch and thus are not included in the example.

In systems with complex displays (such as Air Traffic Control systems), it may also be useful to define various *display modes.*

Output Message Specification : Everything starts from outputs in SpecTRM-RL. By starting from the output specification, the specification reader can determine what inputs trigger that output and the relationship between the inputs and outputs. This relationship is the most critical in understanding and reviewing a system requirements specification, and therefore saliency of this information can assist in these tasks. Other state-machine specification languages, such as RSML and Statecharts, do not explicitly show this relationship, although it can be determined, with some effort, by examining the specification.

An example output specification is shown in Figure D.3. The information included is influenced by the completeness criteria we previously defined for safety-critical blackbox specifications [48, 66]. The completeness criteria play an important role in the hazard analysis process described in Chapter 9.

The following information can and should be included in the specification of the output message: **destination** of the output; **acceptable values**; **timing behavior** including any initiation delay or completion deadline along with any required exception-handling behavior if the deadlines cannot be met, output load and capacity limitations, etc.; **feedback information** about how the controller will determine that the output command has been successfully implemented; and the identity of any other output commands that **reverse** this output.

In many systems, it is important to indicate a maximum time in which the output command remains effective before it is executed. After this time, the output essentially "times out" and should not be executed. This data age information can either be provided in the output message (if the timeout is implemented by the output command actuator) or included in the reversal information if the controller must issue a reversal to undo the command's effect. Reversal information is also useful in identifying accidentally omitted behavior from the specification, i.e., most output actions to provide a change in the controlled plant or supervisory interface have complementary actions to undo that change. **References** are pointers to other levels of the intent specification related to this part of the specification and are used for traceability.

Some of this information may not be applicable or important for a particular system. However, by including a place for it in the specification language syntax, the specifier must either include it or indicate that the information is not applicable or important. The implementers and maintainers need to know that these behaviors are not important and why not and also need to know that it was considered by the original specifiers and not simply forgotten. Lots of accidents and incidents result from such a lack of consideration of these factors by designers and implementers.

The conditions under which an output is triggered (sent) can be specified by a predicate logic statement over the various states, variables, and modes in the specification. In our experience in specifying complex systems, however, we found that the triggering conditions required to accurately capture the requirements are often extremely complex. We also found propositional logic notation did not scale well to complex expressions in terms of readability and error-proneness. To overcome this problem, we developed a tabular representation of disjunctive normal form (DNF) that we call AND/OR tables.

The far-left column of the AND/OR table lists the logical phrases of the predicate. Each of the other columns is a conjunction of those phrases and contains the logical values of the expressions. If one of the columns evaluates to *true*, then the entire table evaluates to *true*. A column evaluates to *true* if all of its elements match the truth values of the associated predicates. An asterisk denotes "don't care."

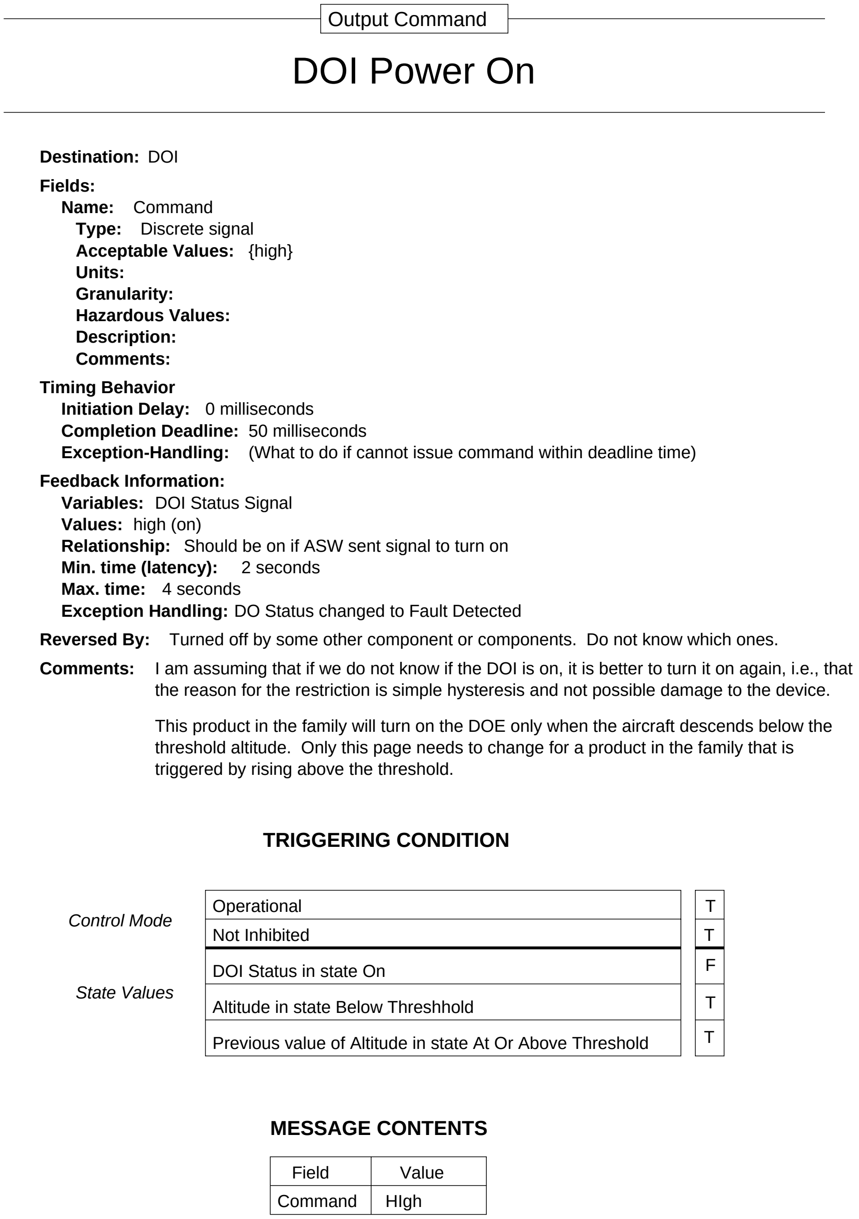
For SpecTRM-RL, we kept the very successful AND/OR tables, but made one addition based on human factors considerations. We now recommend that the output condition tables be organized into two parts: an upper part denoting the relevant control modes for that column and a lower part describing any additional conditions for triggering the output. We have found that this separation assists in completeness checking, particularly when humans are writing and reviewing specifications. For completeness reasons, every output command column must include a reference to the control mode(s) under which the command is sent. It is assumed that if a particular mode is not specified, then the output cannot occur in that mode.

For the altitude switch *DOI-Power-On* output in the example shown in Figure D.3, the command is triggered (sent) when all the following conditions are true: the altitude switch is in the operational and not-inhibited modes, the DOI is not on, the altitude is below the threshold, and the previous altitude was at or above the threshold (the requirements for the altitude switch say that if the switch is turned off while the aircraft is below the threshold altitude, the DOI is not powered on again until the aircraft goes above the threshold altitude and again passes down through it). The *Prev* built-in function, which is a feature of the underlying formal RSM model, allows referring to previous values of modes, state variables, inputs, and outputs. References to time are also allowed in the specification of trigger conditions. An example is shown later.

Input Definition : Our desire to enforce completeness in the language itself (to satisfy our completeness requirements) leads to language features that allow the inclusion of information (if relevant) about input arrival rates, exceptional-condition handling, data-age requirements, etc. No input data is good forever; after some point in time it becomes obsolete and should not be used. We provide a special value, obsolete, that an input variable assumes a specified time after the last value is received for that variable.

In the example shown in Figure D.4, the specification states that the value comes from the *altitude* field in the DA1-message and is assigned when a message arrives. If no message has arrived in the past 2 seconds, the previous value is used. If the last message arrived more than 2 seconds before, the data is considered obsolete. The input variable also starts with the obsolete (undefined) value upon startup. Because of the similarity of the form of most input definitions, we may simplify the notation in the future.

When the controller has multiple supervisory modes, these must be specified to denote which inputs should be used at any particular time.



Output Command

DOI Power On

Destination: DOI

Fields:
Name: Command
Type: Discrete signal
Acceptable Values: {high}
Units:
Granularity:
Hazardous Values:
Description:
Comments:

Timing Behavior
Initiation Delay: 0 milliseconds
Completion Deadline: 50 milliseconds
Exception-Handling: (What to do if cannot issue command within deadline time)

Feedback Information:
Variables: DOI Status Signal
Values: high (on)
Relationship: Should be on if ASW sent signal to turn on
Min. time (latency): 2 seconds
Max. time: 4 seconds
Exception Handling: DO Status changed to Fault Detected

Reversed By: Turned off by some other component or components. Do not know which ones.

Comments: I am assuming that if we do not know if the DOI is on, it is better to turn it on again, i.e., that the reason for the restriction is simple hysteresis and not possible damage to the device.

This product in the family will turn on the DOE only when the aircraft descends below the threshold altitude. Only this page needs to change for a product in the family that is triggered by rising above the threshold.

TRIGGERING CONDITION

Control Mode	Operational	T
	Not Inhibited	T
State Values	DOI Status in state On	F
	Altitude in state Below Threshhold	T
	Previous value of Altitude in state At Or Above Threshold	T

MESSAGE CONTENTS

Field	Value
Command	HIgh

Figure D.3: Example of an Output Specification

Input Value

DA1 Alt Signal

Source: Digital Altimeter 1

Type: integer

Possible Values (Expected Range): -20..2500

Exception-Handling: Values below -20 are treated as -20 and values above 2500 as 2500

Units: feet AGL

Granularity: 1 foot

Arrival Rate (Load): one per second average

Min-Time-Between-Inputs: 100 milliseconds
Max-Time-Between-Inputs: none

Obsolescence: 2 seconds

Exception-Handling: Assumes value Obsolete

Description:

Comments:

Appears in: Altitude

DEFINITION

= New Data for DA1 Alt Signal

DA1 Alt Signal was Received	T

= Previous Value of DA1 Alt Signal

DA1 Alt Signal was Received	F
Time Since DA1 Slt Signal was last received > 2 seconds	F

= Obsolete

DA1 Alt Signal was Received	F	*	*
Time Since DA1 Alt Signal was last received > 2 seconds	T	*	*
System Start	*	T	*
DA1 Alt Signal was Never Received	*	*	T

Figure D.4: Example of an Input Specification

State Variable Definition : State variable values are inferred from the values of input variables or from other state variable values. Figure D.5 shows a partial example of a state variable description for the altitude switch.

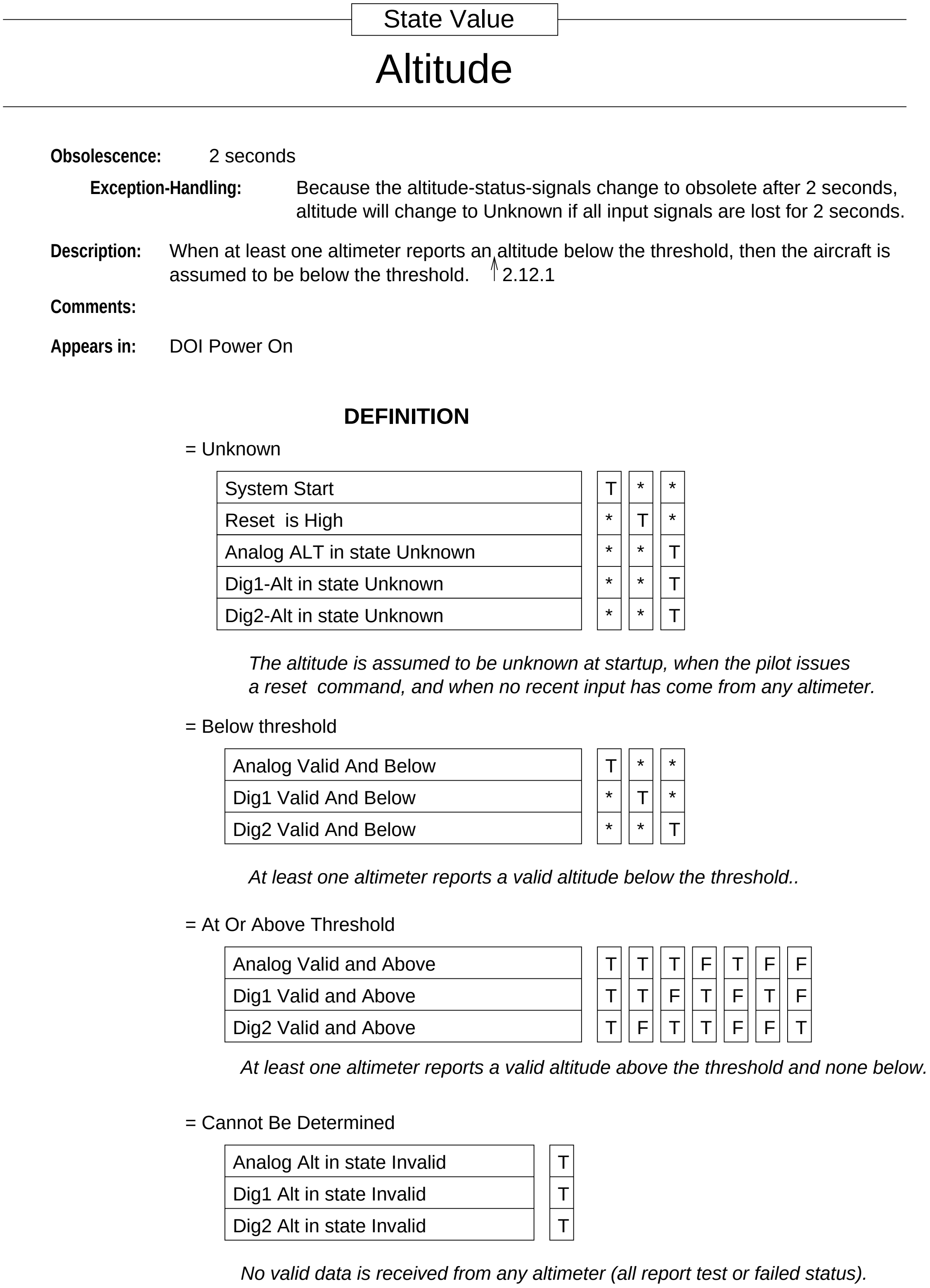
As stated earlier, all state variables that describe the process state should include an *unknown* value. *Unknown* is the default value upon startup or upon specific mode transitions (for example, after temporary shutdown of the computer). This feature is used to ensure consistency between the computer model of the process state and the real process state upon startup or after leaving control modes where processing of inputs has been interrupted. By making *unknown* the default state value and by assuming the unknown value upon changing to a control mode where normal input processing is interrupted (for example, a maintenance mode), the use of an *unknown* state value forces resynchronization of the model with the outside world after an interruption in processing inputs. Many accidents have been caused by the assumption that the process state does not change while the computer is idle or by incorrect assumptions about the initial value of state variables on startup or restart.

If a model of a supervisory display is included in the specification, unknown is used for state variables in the supervisory display model only if the state of the display can change independently of the software. Otherwise, such variables must specify an initial value (e.g., blank, zero, etc.) that should be sent when the computer is restarted.

Macros and Functions : *Macros*, although not strictly necessary, were added to the language for human factors considerations. They are simply named pieces of AND/OR tables that can be referenced from within another table. For example, the macro in Figure D.6 is used in the definition of the state variable *altitude* in the altitude switch example. Its use simplifies the specification of altitude and thus makes it easier to understand while also simplifying making changes and enhancing specification reuse. Macros, for the most part, correspond to typical abstractions used by application experts in describing the requirements and therefore add to the understandability of the specification. In addition, we have found this feature convenient for expressing hierarchical abstraction and enhancing hierarchical review and understanding of the specification. For very complex models (e.g., a flight management system), we have found that macros are an important tool for humans to be able to handle the complexity involved in constructing the specification.

Rather than including complex mathematical functions directly in the transition tables, functions may be specified separately and referenced in the tables. In addition, functions are used in output specifications to define the value computed for the output (e.g., the differential equation used to define a control law).

The macros and functions, as well as other features of SpecTRM-RL, not only help structure a model for readability, they also help organize models to enable specification reuse. Conditions commonly used in the application domain can be captured in macros and common functions can be captured in reusable functions. Naturally, to accomplish reuse, care has to be taken when creating the original model to determine what parts are likely to change and to modularize these parts so that substitutions can be easily made.



State Value

Altitude

Obsolescence: 2 seconds

Exception-Handling: Because the altitude-status-signals change to obsolete after 2 seconds, altitude will change to Unknown if all input signals are lost for 2 seconds.

Description: When at least one altimeter reports an altitude below the threshold, then the aircraft is assumed to be below the threshold. ↑2.12.1

Comments:

Appears in: DOI Power On

DEFINITION

= Unknown

System Start	T	*	*
Reset is High	*	T	*
Analog ALT in state Unknown	*	*	T
Dig1-Alt in state Unknown	*	*	T
Dig2-Alt in state Unknown	*	*	T

The altitude is assumed to be unknown at startup, when the pilot issues a reset command, and when no recent input has come from any altimeter.

= Below threshold

Analog Valid And Below	T	*	*
Dig1 Valid And Below	*	T	*
Dig2 Valid And Below	*	*	T

At least one altimeter reports a valid altitude below the threshold..

= At Or Above Threshold

Analog Valid and Above	T	T	T	F	T	F	F
Dig1 Valid and Above	T	T	F	T	F	T	F
Dig2 Valid and Above	T	F	T	T	F	F	T

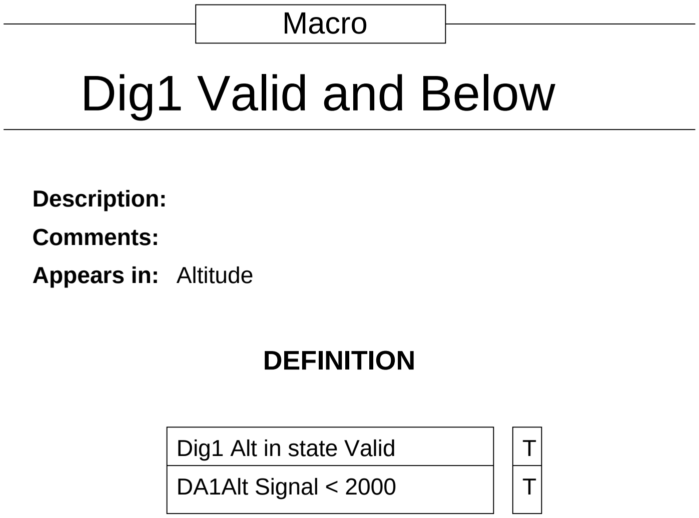
At least one altimeter reports a valid altitude above the threshold and none below.

= Cannot Be Determined

Analog Alt in state Invalid	T
Dig1 Alt in state Invalid	T
Dig2 Alt in state Invalid	T

No valid data is received from any altimeter (all report test or failed status).

Figure D.5: Example of an Inferred State Variable Specification



Macro

Dig1 Valid and Below

Description:

Comments:

Appears in: Altitude

DEFINITION

Dig1 Alt in state Valid	T
DA1Alt Signal < 2000	T

Figure D.6: Example of a Macro Specification

Appendix E

A Brief Introduction to System Dynamics Modeling

SpecTRM-RL, like most models, describes the static structure of systems. In order to handle the organizational components of open systems and the adaptation and changes that occur over time for such human-centered systems, we needed to be able to model dynamic system aspects. For this we used system dynamics models [110].

By focusing on the events immediately preceding accidents, event chains treat a system as a static, unchanging structure. But systems and organizations continually experience change and adaptation to existing conditions. Systems dynamics models are one way to describe dynamic change in systems. They have been used to examine the potential undesired consequences of organizational decision-making.

As noted in Part I of this book, a system's defenses or safety controls may degrade over time due to changes in the behavior of the components of the safety control loop. The reasons for the migration of the system toward a state of higher risk will be system specific and can be quite complex. In contrast to the usually simple and direct relationships represented in event-chain accident models, most accidents in complex socio-technical systems involve relationships between events and human actions that are highly non-linear, involving multiple feedback loops. The prevention of accidents in these systems therefore requires an understanding not only of the static structure of the system (the *structural complexity*) and of the changes to this structure over time (the *structural dynamics*), but also the dynamics behind these changes (the *behavioral dynamics*). SpecTRM-RL models capture the static control structure and are useful in performing a hazard analysis that examines complex control structures and the dynamics or structural changes (failures and dysfunctional interactions) that occur over time in these structures, but not the behavioral dynamics, i.e., the dynamic processes behind these structural changes. To model the behavioral dynamics, we are adapting the modeling techniques of system dynamics to model and understand the dynamic processes behind the changes to the static safety control structure: how and why the safety control structure might change over time, potentially leading to ineffective controls and unsafe or hazardous states.

The field of system dynamics, created at MIT in the 1950's by Jay Forrester, is designed to help decision makers learn about the structure and dynamics of complex systems, to design high leverage policies for sustained improvement, and to catalyze successful implementation and change. System dynamics provides a framework for dealing with dynamic complexity, where cause and effect are not obviously related. It is grounded in the theory of non-linear dynamics and feedback control, but also draws on cognitive and social psychology, organization theory, economics, and other social sciences [110]. System dynamics models, like SpecTRM-RL are formal and can be executed. The models and simulators help to capture complex dynamics and to create an environment for organizational learning and policy design. The combination of STPA, SpecTRM-RL and systems dynamics models provides an extremely powerful integrated risk management approach that goes far beyond what is possible using current techniques.

System dynamics is particularly relevant when analyzing system accidents. The world is dynamic, evolving, and interconnected, but we tend to make decisions using mental models that are static, narrow, and reductionist. Thus decisions that might appear to have no effect on safety—or even appear to be beneficial—may in fact degrade safety and increase risk. System dynamics makes it possible, for example, to understand and predict instances of policy resistance or the tendency for well-intentioned interventions to be defeated by the response of the system to the intervention itself. In related but separate research, Marais and Leveson are working on defining archetypical system dynamic models often associated with accidents to assist in creating the models for specific systems [76].

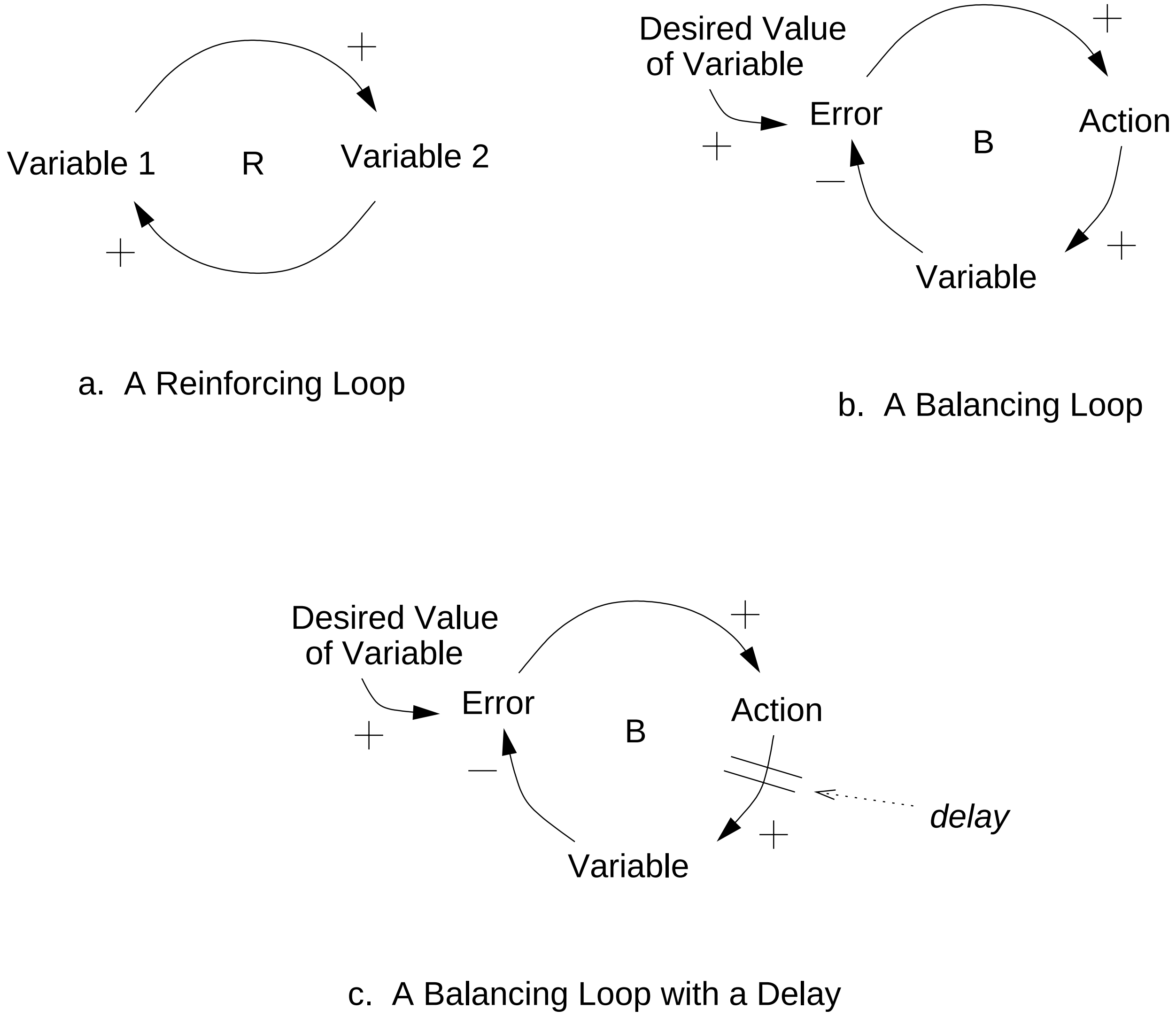
System behavior in system dynamics is modeled by using feedback (causal) loops, stock and flows (levels and rates), and the non-linearities created by interactions between system components. In this view of the world, behavior over time (the dynamics of the system) can be explained by the interaction of positive and negative feedback loops [104]. The models are constructed from three basic building blocks: positive feedback or reinforcing loops, negative feedback or balancing loops, and delays. Positive loops (called reinforcing loops) are self-reinforcing while negative loops tend to counteract change. Delays introduce potential instability into the system.

Figure E.1a shows a *reinforcing loop*, which is a structure that feeds on itself to produce growth or decline. Reinforcing loops correspond to positive feedback loops in control theory. An increase in variable 1 leads to an increase in variable 2 (as indicated by the "+" sign), which leads to an increase in variable 1, and so on. The "+" does not mean that the values necessarily increase, only that variable 1 and variable 2 will change in the same direction. If variable 1 decreases, then variable 2 will decrease. A "-" indicates that the values change in opposite directions. In the absence of external influences, both variable 1 and variable 2 will clearly grow or decline exponentially. Reinforcing loops generate growth, amplify deviations, and reinforce change [110].

A *balancing loop* (Figure E.1b) is a structure that changes the current value of a system variable or a desired or reference variable through some action. It corresponds to a negative feedback loop in control theory. The difference between the current value and the desired value is perceived as an error. An action proportional to the error is taken to decrease the error so that, over time, the current value approaches the desired value.

The third basic element is a *delay*, which is used to model the time that elapses between cause and effect. A delay is indicated by a double line as shown in Figure E.1c. Delays make it difficult to link cause and effect (dynamic complexity) and may result in unstable system behavior. For example, in steering a ship there is a delay between a change in the rudder position and a corresponding course change, often leading to over-correction and instability.

Accident and hazard analysis of socio-technical systems using STAMP starts with a static SpecTRM-RL model of the safety control structure and then uses system dynamics modeling to predict and explain changes in that structure over time.



a. A Reinforcing Loop

b. A Balancing Loop

c. A Balancing Loop with a Delay

Figure E.1: The Three Basic Components of System Dynamics Models

Appendix F

The Thermal Tile Processing System (TTPS) Class Example

F.1 High-Level Description

(The following description is adapted from *A Mobile Robot System for Ground Servicing Operations on the Space Shuttle* by K. Dowling, R. Bennett, M. Blackwell, T. Graham, S. Gatrall, R. O'Toole, and H. Schempf. The original Tessellator robot was designed as a research project in the Robotics Dept. at CMU with NASA funding. Changes have been made from the original specification in order to satisfy different goals).

The Thermal Tile Processing System (TTPS) is designed to service tiles (the thermal protection system) on the Space Shuttle, thus saving humans from a laborious task that begins within minutes after the Shuttle lands and ends just prior to launch—typically three to four months. Upon landing at either the Dryden facility in California or Kennedy Space Center in Florida, the orbiter is brought to either the Mate–Demate Device (MDD) or the Orbiter Processing Facility (OPF). These large structures provide access to all areas of the orbiters.

The Shuttle 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. There are approximately 17,000 lower surface tiles covering an area that is roughly 25m x 40m.

In the current process, DMES is injected into a small hole in each tile by a handheld tool that pumps a small quantity of chemical into the nozzle. The nozzle is held against the tile and the chemical is forced through the tile by a pressurized nitrogen purge for several seconds. The nozzle diameter is about 1 cm but the hole in the tile surface is about 0.1 cm. The heights range from 290 cm to 400 cm from the floor of the OPF. It takes about 240 person hours to rewaterproof the tiles on an orbiter. Because the chemical is toxic, human workers have to wear heavy suits and respirators while injecting the chemical and, at the same time, maneuvering in a crowded work area. One goal for using a robot to perform this task is to eliminate a very tedious, uncomfortable, and potentially hazardous human activity.

The tiles must also be inspected. By inspecting the tiles more accurately than the human eye, it is hoped that the Thermal Tile Servicing System will reduce the need for multiple inspections. During launch, reentry, and transport, a number of defects can occur on the tiles. These defects are evidenced 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 TTPS has three main parts: a mobile robot, a separate (off-board) computer (called the Workcell Controller) that controls the overall thermal tile processing tasks, and a human operator to monitor and control the other two components.

The mobile robot is designed to inspect each tile and inject the waterproofing chemical. Because there are so many tiles, the robot divides its work area into uniform work spaces, inspecting tiles in each area with as little overlap between work spaces as possible.

Before each inspection shift, the operator enters instructions into the Workcell Controller about shuttle position and inspection sequence. The Workcell Controller workstation creates jobs for the mobile robot and updates other NASA databases after the robot uploads data gathered during the course of the shift. This data includes tile images, records of tiles injected and inspected, and other pertinent job data. In addition, robot status data is used to monitor robot operation.

At the beginning of the shift, the mobile robot is downloaded a job. The job consists of a series of files describing the locations, sequences, target IDs, orbiter parking measurements, etc. The robot then uses a rotating laser to position itself under the shuttle, and the robot's camera locates the exact tile to be inspected. Because the shuttle's belly is not flat, the robot 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.

By comparing the current state of each tile with the state of the tile at previous inspections, the mobile robot 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, the robot's updated tile information is entered into existing NASA databases.

On board, computers control the mobile robot's high-level processing tasks while low-level controllers and amplifiers direct arm and wheel motions. Two more computers control the robot's vision and injection systems.

The mobility and positioning system (MAPS) issues movement commands to the motor controller, which directs the wheel motors on the mobile base. MAPS in turn is controlled either by the operator or an on-board computer called the Tile Servicing Subsystem (TSS). The operator controls robot movement and positioning using a hand-held joystick. The TSS controls robot movement and positioning by providing MAPS with a specification of the destination and route.

F.2 Accident Definition

An accident is an unacceptable loss, as defined by NASA Shuttle program management. Unacceptable losses and their severity levels are:

Level 1:

A1-1: Loss of orbiter and crew (e.g., inadequate thermal protection)

A1-2: Loss of life or serious injury in processing facility

Level 2:

A2-1: Damage to orbiter or to objects in the processing facility that results in the delay of a launch and/or result in a loss of greater than TBD dollars.

A2-2: Injury to humans requiring hospitalization or medical attention and leading to long-term or permanent physical effects.

Level 3:

A3-1: Minor human injury (does not require medical attention or requires only minimal intervention and does not lead to long-term or permanent physical effects)

A3-2: Damage to orbiter that does not delay launch and results in a loss of less than TBD dollars.

A3-3: Damage to objects in the processing facility (both on the floor or suspended) that does not result in delay of a launch nor a loss of greater than TBD dollars.

A3-4: Damage to the mobile robot.

> **Assumption:** It is assumed that there is a backup plan in place for servicing the orbiter thermal tiles in case the TTPS has a mechanical failure and that the same backup measures can be used in the event the robot is out of commission due to other types of damage.

F.3 Safety Policy

General Safety Policy: All hazards related to human injury or damage to the orbiter must be eliminated or mitigated by the system design. A reasonable effort must be made to eliminate or mitigate hazards resulting at most in damage to the robot or objects in the work area. For any hazards that cannot be eliminated, the hazard analysis as well as the design features and development procedures, including any tradeoff studies, used to reduce the hazard level must be documented and presented to the customer for acceptance.

Hazard level will be determined by worst potential severity. Hazards that can result in human injury or damage to the orbiter must be eliminated or mitigated if they are not judged to be physically impossible or they are caused by *physical* conditions that are judged to have a likelihood of occurrence of more than one in a million over a 20 year period. All types of software (logical) errors will be considered to be possible and likelihood arguments cannot be used to reduce safety effort related to those errors. A qualitative evaluation of software-related hazard likelihood is acceptable, but as with quantitative evaluations, must be justified to Shuttle Program management and cannot be based simply on the use of testing and good software engineering processes.

F.4 Hazard List

1. Violation of minimum separation between mobile base and objects (including orbiter and humans).
2. Mobile robot becomes unstable (e.g., could fall over).
3. Manipulator arm hits something.
4. Fire or explosion.
5. Contact of human with DMES.
6. Inadequate thermal protection (e.g., damaged tiles not detected, DMES not applied correctly).
7. Damage to robot.

F.5 Work Area (Environment) Assumptions

EA1: The work areas of the Orbiter Processing Facility can be very crowded. The facilities provide access to all areas of the orbiters through the use of intricate platforms that are laced with plumbing, wiring, corridors, lifting devices, etc. After entering the facility, the orbiters are jacked up and leveled. Substantial structure then swings around and surrounds the orbiter at all sides and at all levels. With the exception of the jackstands that support the orbiters, the floorspace directly beneath the orbiter is initially clear but the surrounding structure can be very crowded.

EA2: The mobile robot must enter the facility through personnel access doors 1.1 m (42”) wide. The layout within the OPF allows a length of 2.5 m (100”) for the robot. There are some structural beams whose heights are as low as 1.75 m (70”), but once under the orbiter the tile heights range from about 2.9 meters to 4 meters. Thus the compact roll-in form of the mobile system must maneuver these spaces and also raise its inspection and injection equipment up to heights of 4 meters to reach individual tiles while still meeting the 1 mm accuracy requirements.

EA3: Additional constraints involve moving around the crowded workspace. The robot must negotiate jackstands, columns, workstands, cables, and hoses. In addition, there are hanging cords, clamps, and hoses. Because the robot might cause damage to the ground obstacles, cable covers will be used for protection and the robot system must traverse these covers.