EXTENDING AND AUTOMATING A SYSTEMS-THEORETIC HAZARD ANALYSIS FOR REQUIREMENTS GENERATION AND ANALYSIS

By

JOHN THOMAS

Submitted to the Engineering Systems Division

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2013

Signature of Author:____

Engineering Systems Division
April 29, 2013

Certified by:____

Nancy G. Leveson
Professor of Aeronautics and Astronautics and Engineering Systems
Thesis Committee Chair

Certified by:____

Joseph M. Sussman
Professor of Civil and Environmental Engineering and Engineering Systems
Thesis Committee Member

Certified by:____

John Carroll
Professor of Organization Studies and Engineering Systems
Thesis Committee Member

Accepted by:____

Oliver L. de Weck
Associate Professor of Aeronautics and Astronautics and Engineering Systems
Chair, Engineering Systems Division Education Committee

[Page intentionally left blank]

To my family and my amazing wife.

You were there for me through it all;
for that I am forever grateful.

[Page intentionally left blank]

Extending and Automating a Systems-Theoretic Hazard Analysis for Requirements Generation and Analysis

John Thomas

ABSTRACT

Systems Theoretic Process Analysis (STPA) is a powerful new hazard analysis method designed to go beyond traditional safety techniques—such as Fault Tree Analysis (FTA)—that overlook important causes of accidents like flawed requirements, dysfunctional component interactions, and software errors. Although traditional techniques have been effective at analyzing and reducing accidents caused by component failures, modern complex systems have introduced new problems that can be much more difficult to anticipate, analyze, and prevent. In addition, a new class of accidents, component interaction accidents, has become increasingly prevalent in today's complex systems and can occur even when systems operate exactly as designed and without any component failures.

While STPA has proven to be effective at addressing these problems, its application thus far has been ad-hoc with no rigorous procedures or model-based design tools to guide the analysis. In addition, although no formal structure has yet been defined for STPA, the process is based on a control-theoretic framework that could be formalized and adapted to facilitate development of automated methods that assist in analyzing complex systems. This dissertation defines a formal mathematical structure underlying STPA and introduces a procedure for systematically performing an STPA analysis based on that structure. A method for using the results of the hazard analysis to generate formal safety-critical, model-based system and software requirements is also presented. Techniques to automate both the STPA analysis and the requirements generation are introduced, as well as a method to detect conflicts between safety requirements and other functional model-based requirements during early development of the system.

Thesis Supervisor: Nancy. G. Leveson

Title: Professor of Aeronautics and Astronautics and Engineering Systems

[Page intentionally left blank]

ACKNOWLEDGEMENTS

"The future pivots around you. Here. Now. So do good. For humanity, and for Earth."
—The Doctor

First and foremost, I must thank my advisor Professor Nancy Leveson for believing in me. If she had not invited me into her research group that first semester, none of this would have been possible. I had a lot to learn, and your inspiration and encouragement were essential in taking those first steps and eventually bringing the ideas in this dissertation to life.

Thanks are also due to my committee members, Professor Joseph Sussman and Professor John Carroll, who offered much needed advice when I was getting started and have since provided invaluable feedback along the way. Without their support, this work would not have been possible.

I owe gratitude to the current researchers in Nancy's group that I work with every day. Cody Fleming, John Helferich, Bill Young, and Takuto Ishimatsu all provided comments and feedback every step of the way, which was immensely helpful in improving and refining the ideas in this dissertation. Blandine Antoine, who finished her dissertation just before me, was especially helpful in offering constructive feedback when these ideas were brand new and was even brave enough to apply some of these ideas to the largest project in our lab at the time. Previous graduate students—Maggie Stringfellow, Matthieu Couturier, Brandon Owens—gave early advice that had a lasting impact and helped shape the direction of this research. For the new students—Cameron, Seth, Dan, Connor, Dajiang, and Ian—thanks for keeping me on my toes with your excellent questions and comments. I also thank Dr. Qi Hommes, who offered support throughout the process.

I need to give special thanks to Francisco Luiz De Lemos, an Engineer at the Nuclear and Energy Research Institute, IPEN/CNEN, in Brazil. Without your expertise, the nuclear examples throughout this dissertation and in the appendix would not exist.

Finally, my greatest appreciation goes to my wife Judy and my family. You supported me from the beginning and without that, I could not have done any of this. I love you all!

[Page intentionally left blank]

TABLE OF CONTENTS

[Page intentionally left blank]

LIST OF FIGURES

LIST OF TABLES

Chapter 1. Introduction and Motivation

The introduction of new technology, such as computers and software, is changing the types of accidents we see today. The level of complexity in many of our new systems is leading to accidents in which no components failed but instead unsafe interactions among non-failed components lead to the loss. At the same time, traditional hazard analysis techniques assume accidents are caused by component failures or faults [1] and oversimplify the role of humans [2, 3]. Attempts have been made to extend these traditional hazard analysis techniques to include software and cognitively complex human errors, but the underlying assumptions do not match the fundamental nature of systems we are building today. For example, most software-related accidents can be traced to incomplete or flawed software requirements [4, 5], however current hazard analysis methods like Fault Tree Analysis (FTA) analyze component failures and easily overlook unsafe requirements. In addition, new technology is changing the role of humans in systems from followers of procedures to supervisors of automation and high-level decision makers [6, 7]. New models of accident causation and hazard analysis techniques are needed to address these issues.

1.1 Challenges in engineering safe systems

In the past, most accidents could be traced to unreliable components or sudden failures in the system. Fortunately, safety engineering techniques today are very effective at preventing accidents caused by component failures. In fact, they have become so effective that most major accidents today are caused not by component failures but by complex and often unexpected interactions among components operating as designed. This is especially true for software components notorious for causing accidents by faithfully executing flawed programming and instructions. While modern software is becoming more and more complex and difficult to analyze, safety-critical systems are growing increasingly dependent on and integrated with software components.

The increasing use of software is also changing the nature of human-computer interaction and operator tasks. In the past, operators often had straightforward tasks with simple and precise instructions. However, more and more systems today employ automation whenever possible to perform these tasks, thereby reducing operator workload and the human resources required. More and more operators have therefore had to adapt to a new role of supervising these advanced computer systems and performing miscellaneous tasks that could not be automated. Instead of executing straightforward procedures and learning simple rule-based behaviors, operators are now responsible for much more complex decision-making such as diagnosing new problems and developing solutions on-the-fly. This trend significantly increases the coupling between man and machine—human errors are no longer dominated by trivial problems like lapses or distractions but by complex processes that depend on the type and quality of feedback humans receive and the context of the system in which they are operating.

1.2 Limitations of current approaches

Current approaches to safety engineering are well-suited to prevent some accident causes, such as component failures, but identifying other causes like flawed requirements is much more difficult. In addition, the most popular approaches are based on a model of components that can misbehave in predictable ways or with well-known failure modes, but complex software and new types of human errors are outgrowing those models. Safety engineering activities are usually restricted to later phases in the development process when a fairly detailed design is available to analyze, but this approach limits the number of solutions available and causes design changes when they are most expensive to implement.

A number of approaches have been developed to analyze complex human tasks [8] or detailed software models [9]. However, these techniques are only applicable to specific system components—humans or software, respectively—and do not provide an integrated view of the overall safety at a system level. Safety is an emergent system property [5, 10] that depends on the behavior of and interactions among many components in the system—in other words, safety is a *system* property that only emerges from the integration of components within the system and the environment. Software, for example, is by definition an idea abstracted from physical realization and cannot by itself be safe or unsafe; software can only cause an accident by affecting the behavior of other components. In fact, many accidents are caused not by any single component but rather from complex processes involving a series of interactions among concurrently operating components. Unfortunately, while various analysis methods exist for individual components, there are very few comprehensive approaches that can be applied to analyze the entire system including complex relationships and interactions between software, hardware, and human components.

System Theoretic Process Analysis (STPA) is a promising approach that was designed to analyze safety in socio-technical systems with many diverse components interacting together [10]. Rather than beginning with an assumption of known and predictable component behavior, STPA derives the necessary safety constraints from the top down in progressive levels of detail. STPA has been successful at identifying a wide range of potential accident causes, including flawed software requirements and complex human interactions. However, STPA is relatively new and its application has thus far been ad-hoc with no rigorous procedures, formal mathematical definition, or model-based design tools to aid in analyzing complex human- and software-intensive systems.

1.3 Research goals and outline

The goal of this dissertation is to advance the state of the art by defining a formal mathematical structure underlying STPA and by creating procedures for systematically performing an STPA analysis. Chapter 2 reviews hazard analysis techniques and discusses the underlying assumptions and implications for complex human- and software-intensive systems. Chapter 3 proposes extensions to STPA including a new procedure for identifying unsafe control actions and causal factors. Chapter 4 provides formal mathematical definitions and presents automated methods to assist in identifying hazardous control actions during early system development as well as automatically generating or validating formal model-based safety requirements. An algorithm for automated conflict detection using the results of the hazard analysis is also discussed, as well as application to non-safety-related functional goals of the system. Chapter 5 discusses the scalability of these extensions and proposes several techniques to manage the complexity of large-scale systems.

[Page intentionally left blank]

Chapter 2. Literature review and evaluation

This chapter briefly reviews traditional system safety approaches including models and analysis techniques and evaluates current approaches with respect to their ability to ensure safety in complex systems. This is followed by a broader evaluation of failure-based models in general and a discussion of the need for a more powerful systems-based approach to safety.

2.1 Accident Models

All safety efforts, whether prospective or retrospective, are based on some *accident model* that represents the theory of accident causation used to analyze or design a system [10]. An accident model includes a set of assumptions about how accidents arise, what factors can lead to accidents, and how those factors combine, interact, or propagate to cause an accident. In practice, an accident model may be a model taught during formal education/training or it may be a mental model that exists only in the mind of the analyst or engineer, perhaps formed over years of experience. In either case, knowing the accident model that is used and recognizing the underlying assumptions is the first step to understanding the strengths and limitations of any safety analysis.

This section describes various accidents models in use today and summarizes their assumptions and limitations. The next section describes analysis methods that use accident models to identify potential accident scenarios and causes in a given system.

2.1.1 Domino Accident Model

Herbert Heinrich published one of the earliest accident models in 1931, known as the Domino Accident Model [11]. Heinrich worked for Travelers Insurance Company, where he was exposed to countless industrial accident reports. Although ad-hoc analyses were common, Heinrich believed that most investigations were too superficial and provided insufficient information to identify root causes and prevent future accidents. Heinrich believed that the vast majority of industrial accidents—98 percent—were preventable if the “true causes” could be identified using a better model of accident causation.

At the time, industrial accidents were often traced to faulty machinery or design flaws such as inadequate safety guards on industrial machines. However, Heinrich believed that as more complex industrial machinery was introduced and safety features became common, a new cause was increasingly to blame: “man failure”. As he explains:

> *Mechanization of industry created new machines and new dangers. At first these additional hazards received little attention, and the greater number of industrial accidents were caused by the use of unguarded mechanical equipment. Under these conditions it was perfectly proper to charge accidents to specifically named machines, parts of machines, or mechanical equipment, and to state that these things were the actual causes of accidents. With the passing of time, important changes took place in industry. Improved equipment and methods were introduced, better guards were devised and were more commonly used, accidents from purely mechanical or physical causes decreased, and man failure became the predominating cause of injury. [11]*

> *Faulty instruction, lax discipline, unsafe practices, inattention, and physical and mental impairment or inefficiency are some of the real causes of construction accidents, and they produce 88 per cent of all injuries [11].*

Heinrich’s Domino model, shown in Figure 1, models five stages of an industrial accident. According to the model, accidents are inevitably caused by either unsafe acts of a worker or unsafe conditions in the workplace. Heinrich believed that 88% of accidents were caused by unsafe acts while 10% of accidents were caused by unsafe conditions. These acts and conditions are in turn caused by the fault of a person, which is a result of their ancestry and social environment. Heinrich argued that accident processes were like a series of dominos falling—each event in the sequence automatically causes the next event. Removing any domino would break the chain and prevent accidents. Much of Heinrich’s work focused on identifying and removing the middle domino—especially unsafe acts in an industrial workplace—and he published several classifications of this cause. One such classification is shown in Figure 2.

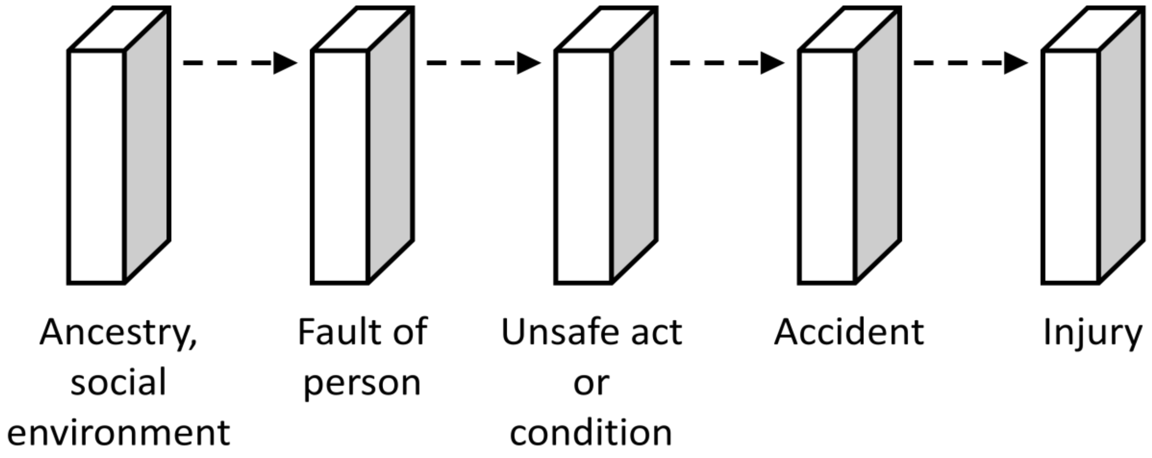


Figure 1: Heinrich's Domino Accident Model

Unfortunately, the data and accident reports Heinrich used were never published. The claims that 98% of accidents are preventable or that 88% of accident causes are due to unsafe human supervision could not be verified. Nevertheless, many find the model intuitive and it quickly became popular. Although the original purpose of the Domino model was to understand operator injuries in an industrial environment, it has been applied in almost every industry and was very influential in shaping subsequent accident models. Several key assertions in the Domino model still persist today, however critics argue that these assertions can lead to serious problems when applied to modern complex systems.

Key assertions in the Domino Accident Model:

1. Accidents are best understood as a chain of events
2. A direct causal relationship exists between events resulting in linear propagation
3. Accidents are primarily caused by a single "root cause" or "proximate cause"
4. Accidents are primarily caused by operator error

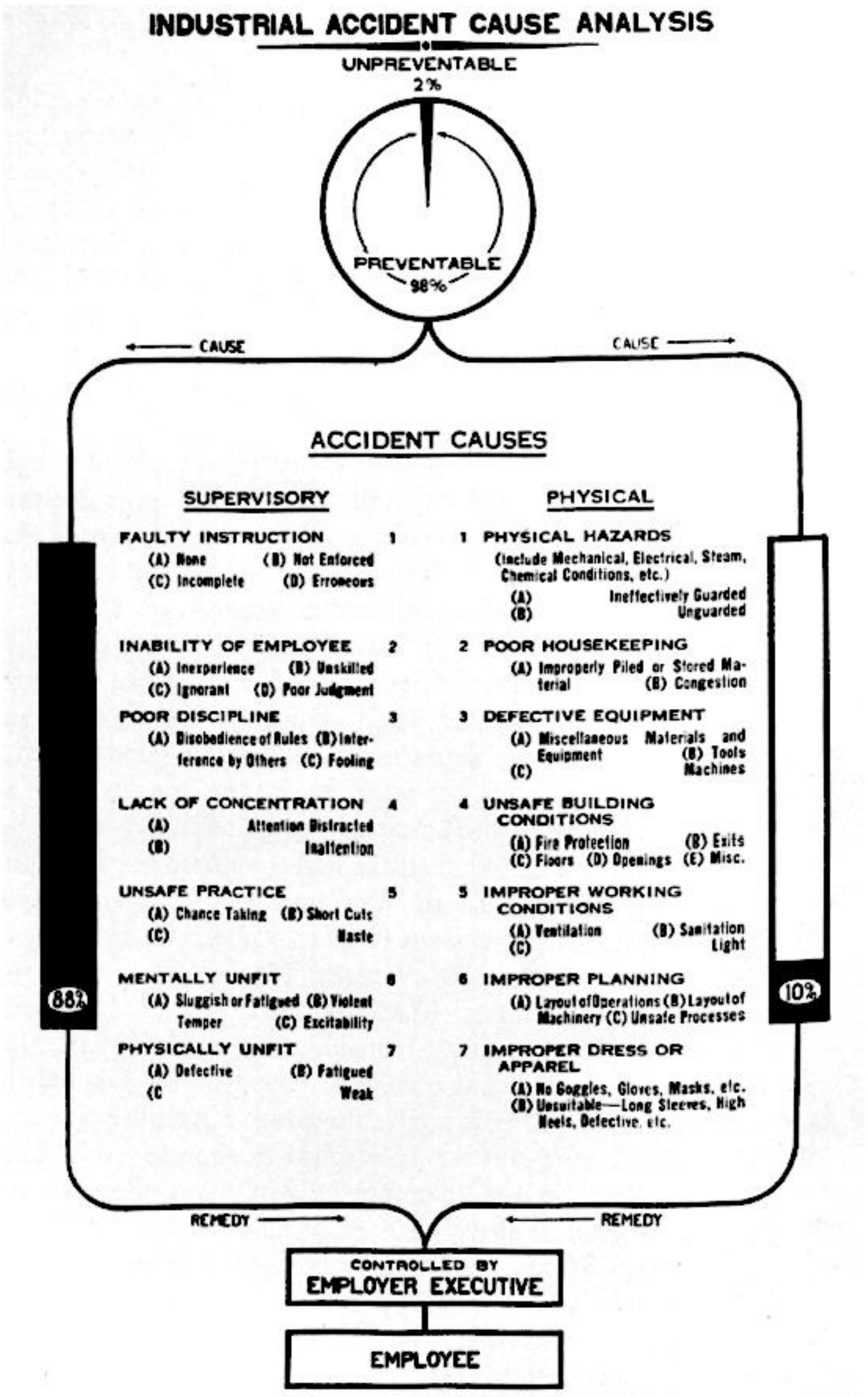

INDUSTRIAL ACCIDENT CAUSE ANALYSIS
UNPREVENTABLE
2%
PREVENTABLE
98%
CAUSE
CAUSE
ACCIDENT CAUSES
SUPERVISORY
PHYSICAL
FAULTY INSTRUCTION 1
(A) None (B) Not Enforced
(C) Incomplete (D) Erroneous
INABILITY OF EMPLOYEE 2
(A) Inexperience (B) Unskilled
(C) Ignorant (D) Poor Judgment
POOR DISCIPLINE 3
(A) Disobedience of Rules (B) Interference by Others (C) Fooling
LACK OF CONCENTRATION 4
(A) Attention Distracted
(B) Inattention
UNSAFE PRACTICE 5
(A) Chance Taking (B) Short Cuts
(C) Haste
MENTALLY UNFIT 6
(A) Sluggish or Fatigued (B) Violent Temper (C) Excitability
PHYSICALLY UNFIT 7
(A) Defective (B) Fatigued
(C Weak
1 PHYSICAL HAZARDS
(Include Mechanical, Electrical, Steam, Chemical Conditions, etc.)
(A) Ineffectively Guarded
(B) Unguarded
2 POOR HOUSEKEEPING
(A) Improperly Piled or Stored Material (B) Congestion
3 DEFECTIVE EQUIPMENT
(A) Miscellaneous Materials and Equipment (B) Tools
(C) Machines
4 UNSAFE BUILDING CONDITIONS
(A) Fire Protection (B) Exits
(C) Floors (D) Openings (E) Misc.
5 IMPROPER WORKING CONDITIONS
(A) Ventilation (B) Sanitation
(C) Light
6 IMPROPER PLANNING
(A) Layout of Operations (B) Layout of Machinery (C) Unsafe Processes
7 IMPROPER DRESS OR APPAREL
(A) No Goggles, Gloves, Masks, etc.
(B) Unsuitable—Long Sleeves, High Heels, Defective, etc.
88%
10%
REMEDY
REMEDY
CONTROLLED BY
EMPLOYER EXECUTIVE
EMPLOYEE

Figure 2: Classification of industrial accident causes from Heinrich [11]

Assertion 1: Accidents are best understood as a chain of events

Heinrich's Domino model is one of the earliest examples of a chain of events model. The basic premise is that accidents are caused by events and that events provide the information necessary to prevent accidents. Heinrich argued that once the true cause is identified, the "practicable remedy" for other similar accidents can be expressed as the reverse of the assigned cause. For example, he notes that if the unsafe act in the Domino model is identified as "instruction disregarded" then the appropriate remedy should be to simply "enforce instruction".

Critics argue that this characterization omits many important interacting factors including the operator's understanding of the machine, past experiences with other machines or supervisors, cognitive learning processes, and potentially conflicting goals. In fact, instructions can be disregarded for good reasons and the best remedy may actually involve adopting a more intuitive procedure rather than increasing the enforcement of a poor procedure. Moreover, the reason for a disregarded instruction may not be due to an inferior operator *or* procedure but due to a system design in which proper operation is cumbersome or may cause unexpected problems elsewhere. In addition, factors such as the organizational safety culture, management pressure, goal prioritization, and financial incentives can all play critical roles in influencing the events that unfold even though none of these factors is an actual event. These facts have been found again and again not only in industrial workplaces, but also in modern systems across many industries including aviation, space systems, ground transportation, and nuclear power systems [10]. The complex causes behind many of today's accidents can only be found by exploring beyond a chain of events; accident prevention requires consideration of many interacting factors that are easily omitted in a chain of events model.

Assertion 2: A direct causal relationship exists between events resulting in linear propagation

The Domino model is also an example of a linear model: each event propagates automatically in a direct causal manner to the next event in the linear time-ordered sequence. Given that the accident occurred, each domino in the model is treated as necessary and sufficient to cause the subsequent domino to fall [5, 11].

However, linear models only capture one-way dependencies and do not include non-linear interactions and indirect causation such as continuous feedback loops, mutually reinforcing behavior, adaptive processes, changing environments, etc. In reality, accidents are complex situations involving many interdependent causes—each of which may be insufficient and not necessary to cause an accident—but in combination they can produce disastrous and often unanticipated results. Events do not always propagate automatically in a neat linear fashion; an action or behavior that did not cause an accident yesterday may very well cause an accident today. Critical factors can influence, but not directly cause, the events leading to an accident. Operators and engineers learn and adapt not only from recent events, but also from previous accidents (modeled in event chains as an outcome only), from other past experiences, and from collaboration with each other. Actions in software or human systems are rarely purely unidirectional; they are often cooperative in nature and may involve several feedback loops and iterations before an accident occurs. Although linear models are intuitive and easy to apply, they do not match the fundamental nature of accidents in today's complex systems.

Assertion 3: Accidents are primarily caused by a single "root cause" or "proximate cause"

An important consequence of a linear propagation model is that the outcome can be easily prevented in the model if any single event in the chain is prevented—i.e. by breaking the chain. As a result, linear propagation models tend to place emphasis on a single "root cause" that is easy to prevent while overlooking other contributory causes. Although Heinrich admits that some accidents have multiple causes, he argues that accidents still can and should be traced to a single primary or proximate cause for the purpose of preventing accidents. Heinrich praises the U.S. Bureau of Labor Statistics for providing a "clear and satisfactory definition" of proximate cause as follows:

> *The accident should be charged to that condition or circumstance the absence of which would have prevented the accident; but if there be more than one such condition or circumstance, then to the one most easily prevented. [12]*

Heinrich adopts this definition, but also advocates an alternative criterion. By following a process of elimination, Heinrich proposes starting with the most common causes until one is found that in the analyst's opinion is responsible for the accident. Heinrich refers to this cause as both the "real cause" and the "proximate cause" and argues that the selection is not arbitrary but an obvious and demonstrable fact that in most cases is easily determined.

> *The person who is to select the cause must know all the pertinent facts. With these in mind, he may follow a process of elimination, beginning by considering the most common and most readily attacked causes. If he has already decided that machine guarding was not at fault, nor did congestion or inadequate light, lack of protective appliances, or other possible physical causes exist, he may proceed to eliminate the moral causes. He should find out whether safety instructions were required; if so, whether they had been given and enforced, or disobeyed and disregarded by the employee. If so far is satisfactory, he should proceed to the next item in the list of true cases and so exhaust the various probabilities until he strikes the cause which in his opinion is responsible. Laborious at first, practice shortly enables rapid diagnosis to be made. Having assigned a tentative cause, the test of reversal as explained in the first part of this chapter is applied, and if by this method a practicable remedy is evolved, the task of cause-analysis is completed [11].*
>
> *If an employee, for example, contrary to instructions and with knowledge of the hazard, deliberately removes a guard from a set of gears and gets hurt, it is obvious that a moral rather than a physical cause of the accident existed and, more specifically, that it was disregard of instruction. Certainly this is the obvious proximate, real cause—one that can be attacked and that should be attacked if recurrence is to be avoided. Moral fault in many cases is a demonstrable fact which may easily be determined [11].*
>
> *If the principles advocated in this book are followed, analysis will reveal the first immediate or proximate real cause of such accidents [11].*
>
> *Stated in another way, the application of sound principles not only shows what to attack but also indicates what not to attack; and, again, by this process of eliminating non-essentials, focuses attention upon a major remedy in the control of the frequency and severity of injuries [11].*

Although identifying a single primary cause greatly simplifies the analysis effort, doing so omits other important causes and is contrary to the goal of preventing accidents. Limiting the identified causes severely limits the potential solutions, while identifying more causes allows several potential solutions to be readily identified—including solutions that may be more flexible, more effective, and less expensive. Moreover, the determination of a single cause is always arbitrary and dependent on opinion rather than any demonstrable fact. In Heinrich's own example, the declaration that "moral fault" and "disregard of instruction" is the real cause is not obvious and overlooks many equally important causes including machines in which guard removal is necessary to oil the gears, a lack of interlocks to prevent unsafe guard removal (whether intentional or not), and managerial pressures and financial incentives to perform maintenance without interrupting production. Although linear models suggest that in theory only one factor needs to be addressed to break the chain, in reality it is only a matter of time before these other unchecked hazardous conditions can lead to an accident. Linear models can also lead to overconfidence through an oversimplified analysis based on a few very specific accident scenarios while in reality an infinite number of accident scenarios are possible; preventing a single event may not actually prevent or even delay an accident. In fact, any one of these hazardous conditions could be called an "accident waiting to happen". If the goal is to prevent future accidents, then such hazardous conditions cannot be omitted outright from consideration as potential targets for correction.

In practice, the ascription of primary cause is often influenced by other qualities—whether intentional or not—including legal liability and the cost to resolve it. Unfortunately, the cheapest solutions are often the least effective in preventing accidents, and organizations may be more likely to blame factors outside their responsibility for legal and other reasons. In Heinrich's example, reprimanding the employee who is already injured may be the cheapest response, but if the conditions that prompted his action are never addressed then the accident is likely to recur with the same or a different worker.

The tendency to assign blame to a small number of causes is profound. In the recent Toyota unintended acceleration debacle, Toyota and the National Highway Traffic Safety Administration primarily blamed the cause on dealerships who installed oversize floor mats, accelerator pedals that became stuck, or driver error [13, 14]. However, many other causes were just as important including the inability to brake under engine power, a design that permitted simultaneous braking and acceleration, and a keyless electronic ignition that by design ignored user attempts to turn off the engine while driving.

During the 2010 Deepwater Horizon oil spill, British Petroleum (BP) argued that the primary accident causes involve failures of the Transocean rig crew and unstable cement provided by Halliburton [15, 16]. Meanwhile, Transocean insisted BP's risky but cost-saving well design was to blame and Halliburton argued that BP's specifications were at fault [17, 18]. BP's internal report eventually identified eight primary causes, however Senator Ed Markey noted that "of their own eight key findings, they only explicitly take responsibility for half of one." [19]

This problem is not new. In 1972, a DC-10 aircraft baggage door blew out during flight, collapsing part of the floor and severing cables and hydraulics used to control the aircraft [20]. The manufacturer found that the proximate cause was improper baggage handler procedures that used extra force to close the door. Baggage handlers were advised of proper procedures, however many other causes were ignored including a door design that closed without latching, a handle that appeared locked when it wasn't, a cockpit light that incorrectly indicated the door was secure, the lack of a vent to prevent floor collapse during depressurization, and redundant control lines that all ran along the section of floor that would collapse during depressurization [20-22]. Despite all the emphasis on one small part of the event chain—baggage procedures—the other hazardous conditions remained and the accident happened again two years later when a second DC-10 baggage door blew out during flight. This time the crash was fatal, killing all 346 people aboard [22].

In each case, the conclusions were based on an implicit linear chain of events model that focuses on proximate causes as Heinrich and others proposed nearly a century ago. The selection of the proximate cause or causes is clearly not free from bias or opinion, and in many cases the selection only masks many other equally important underlying causes that must be identified and examined if we are to prevent accidents. Although many find linear models intuitive at first and may not give much thought to the underlying assumptions and implications, in reality these factors deserve careful evaluation before any conclusions are drawn.

Assertion 4: Accidents are primarily caused by operator error

Heinrich not only believed that most accidents have a primary cause—he argued that the primary cause is almost always human error. Although progress had been made in terms of improved safety features like machine guards to protect from powerful gears or other dangerous components, accidents still happened. Heinrich noted that in almost every accident—whether or not safety features were present—an alternative human behavior could usually be identified that would have prevented the accident. Heinrich believed that by simply enforcing proper behavior and procedures, 88% of all industrial accidents could be prevented. This approach was notably different from the conventional practice at the time that primarily identified causes rooted in the physical environment such as hazardous conditions or design flaws that could be corrected.

Heinrich was very influential in shifting safety efforts from designing machines and procedures that better accommodate human behavior to a renewed focus on adapting human behavior to better accommodate the existing physical environment and enforcing strict supervision. As a result, blame for accidents began to shift from poor workplace conditions and poorly designed machines to human and supervisory errors, or as Heinrich called it, "man failure". He writes:

> *The causes enumerated … are the first, true, proximate or immediate causes most readily and practicably eliminable. A point of real distinction, however, is that heretofore it has been thought possible to apply such a definition only to purely physical or mechanical conditions (for example, unguarded machinery) and not to moral conditions or man failure; whereas, research, experimentation, and practice now prove conclusively that the generality of "man failure" may be broken down just as readily into specific causes of accidents [11].*
>
> *It was discovered that 25 per cent of all accidents would, according to the usual improper method of analysis, be charged to plant physical or mechanical causes, but that in reality the causes of many accidents of this type were either wholly supervisory or chiefly supervisory and partly physical. This group, therefore was found to be actually 10 instead of 25 per cent. This difference (15 per cent) added to the 73 per cent of causes that are obviously of a supervisory nature, gives a total of 88 per cent of all industrial accidents that can be prevented through the enforcement of proper supervision.*
>
> *[regarding a fatal accident] Employee opened the door of a gear guard to oil the machinery. His arm was drawn into the heavy gears. Instructions had been issued not to open the door of a guard while the machinery was in motion. … The gear guard was so*

> *built that the door could readily be opened. The oil cup was located inside the guard instead of outside. … There is no question as to the primary cause. Disregard of instructions was deliberate. An employee of this type would find ways to get hurt even on well-guarded equipment [11].*[1]
>
> *Employees are continually being hurt because they do unsafe things or omit to take safety precautions. The immediate practical remedy must be based on knowledge of what these dangerous practices are, of the first most readily corrected cause, and on supervisory observation and pressure in correction. If the cause should, for example, be non-enforcement of instruction, it is not always necessary to find out why the instruction is not enforced. …. Since it is not always necessary to delve so deeply into the origin of accidents, it is clear that for all practical present purposes a line of demarcation may be drawn after the first of the chain of real causes and subcauses has been found [11].*

Critics argue that focusing primarily on human contributions to accidents omits many other possible causes and results in a limited number of solutions for accident prevention [5, 23]. More importantly, human behavior is intimately connected to the context in which it is made; human error is usually a symptom of deeper trouble and the beginning of an inquiry, not a conclusion [2, 3]. Because decisions are rarely made without any reason, preventing unsafe human behavior requires understanding the reason behind potentially unsafe actions. In Heinrich's example above, reasons for opening the door during operation may include the ease of opening the door, oil cup located inside the guard instead of outside, production pressures, contradicting instructions, past experiences, observations of other workers, cumbersome procedures, and organizational culture.

Even if the employee is incompetent, there is still the question of what inadequate policies and procedures led to hiring an incompetent employee and placing him in command of a lethal machine. If the employee is merely fired (or in this case, killed) without addressing any of these issues, the risk of similar future accidents remains; the hazardous conditions and risks caused by deficiencies

[1] Here Heinrich himself noted some of the other factors involved, but still he insisted that man-failure was unquestionably the primary cause that deserves attention in preventing similar future accidents using the Domino model.

in the system are not eliminated, they are merely applied to a new employee. Nevertheless, linear models like the Domino model suggest that it is only necessary to identify and classify the unsafe act with very little analysis of the environment or context that might permit the unsafe act or, worse, make it appear safe and reasonable. In practice, simply admonishing workers and reissuing existing instructions while ignoring the underlying causes—such as inadequate information available, conflicting goals, external pressures, etc.—has not been very successful.

Critics also note that many potential human acts in the Domino model may not always be undesirable. Because of Heinrich's experience at an insurance company, his data sample was inherently biased with samples selected based on outcome (accident or insurance claim). In a linear model, every event preceding an accident is a potential cause; there was very little, if any, study of events that did not lead to an accident or actions that prevented accidents. For example, disregard of instruction may appear as an event preceding an accident but the same action may also occur—perhaps more often—without causing an accident. In fact, disregard of instruction may be praised in some cases if it leads to increased efficiency or savings and may even prevent accidents in other cases when unanticipated conditions or unexpected equipment behavior is encountered. In fact, the recommendations suggested by a linear model—e.g. enforce instructions—may lead to new kinds of accidents.

More recent work [24] has found that occasional deviation from procedure is an essential part of human learning and adaptation processes, and that the best approach is often to provide *more* flexibility in the procedures that people follow. By enforcing stricter and more detailed procedures as suggested by a linear model, workers' ability to learn and adapt may be impaired, limiting their ability to effectively prevent accidents in the future. Moreover, the focus on proximate human acts tends to overemphasize certain behaviors in the narrow context of a single accident scenario while discounting the effects in other situations. The result is often a set of overly generalized conclusions and rigid procedures that may only be valid in a few specific situations.

2.1.2 Extended Domino Models

Heinrich's Domino model was revised twice in 1976 to provide additional emphasis on certain management issues. Bird and Loftus [25] proposed renaming the five stages as follows:

1. Lack of control by management
2. Basic causes (personal and job factors)
3. Immediate causes (substandard practices/conditions/errors)
4. Accident or Incident
5. Loss

Adams [26] proposed the following stages instead:

1. Management structure (objectives, organization, and operations)
2. Operational errors (management or supervisor behavior)
3. Tactical errors (caused by employee behavior and work conditions)
4. Accident or incident
5. Injury or damage to persons or property

Both proposals suggest consideration of management rather than ancestry and social factors, but neither proposal deviated from the fundamental linear structure of the model or the underlying assumptions. Both retained the basic 5-stage model and inherited the same major deficiencies of the original Domino model described above.

2.1.3 Swiss Cheese Accident Model

The Swiss Cheese model was proposed by James Reason in 1990 [27], and it has become increasingly popular in aviation [28] and healthcare [29]. Reason argued that accidents were caused by failures in four stages: organizational influences, unsafe supervision, preconditions for unsafe acts, and unsafe acts. Each stage can be represented by a slice of Swiss cheese and the holes in the cheese represent a failed or absent defense in that layer. Figure 3 illustrates this model.

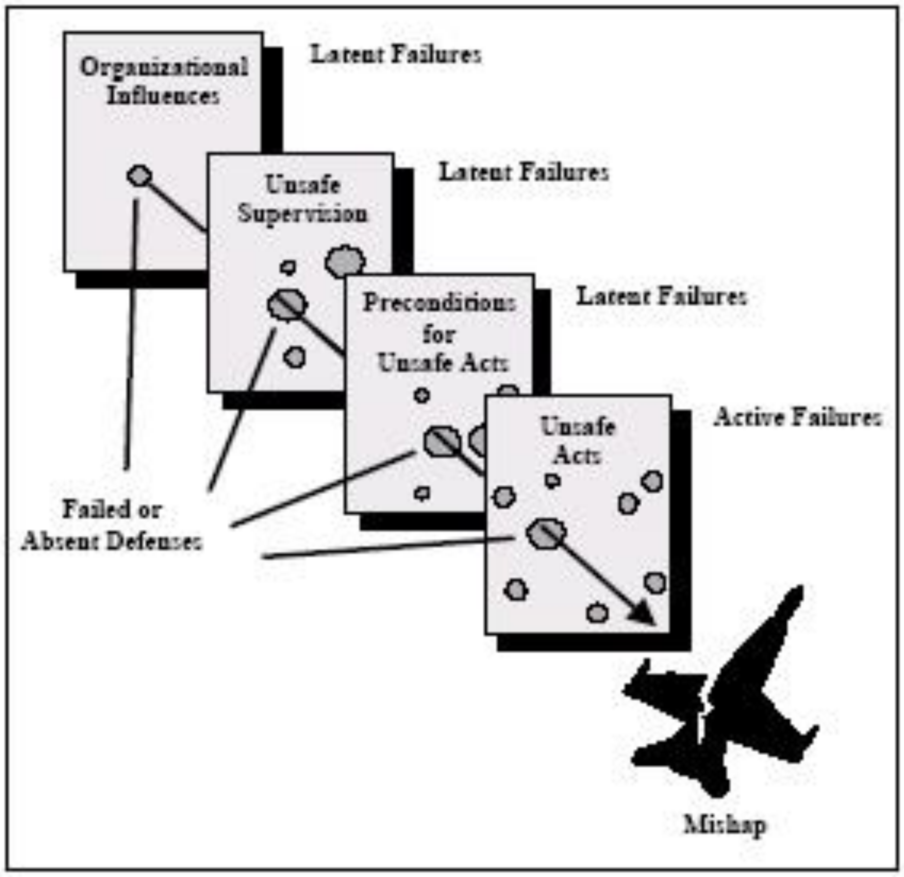


Figure 3: Reason’s Swiss Cheese Accident Model [27, 30]

Reason argues that the holes representing individual weaknesses are randomly varying in location and size. Eventually the holes come into alignment so that a trajectory is possible through all slices, representing a sequence of failures throughout several layers of defense. Failures then propagate along the trajectory through each defense barrier and cause an accident.

The Swiss Cheese model includes many features of the Domino model—including the linear chain of events structure—and suffers from many of the same weaknesses. However, the Swiss Cheese model also relies on two new assumptions: random behavior of components and independence between failures in each layer. Critics argue that these assumptions do not hold in practice, especially for safety-critical software and human behavior [3, 31, 32]. Both software and human behavior are governed by human perception and understanding of the system and environment, which is not random. In fact, most software-related accidents are caused not by random programming mistakes but by systematic flaws in requirements produced by humans [10, 33-35]. For the same reason, the behavior of system components is usually not truly independent in practice. Many important systemic factors can simultaneously affect the behavior of multiple layers—including flawed requirements, inadequate communication, past experiences, etc. While the Swiss Cheese model tends to focus safety efforts on individual holes or failures, the implication of randomness and independence overlooks processes that create holes in every layer, accidents that can occur without any failures, and systemic factors that cause the system as a whole to migrate towards states of high risk.

Like the Domino model, the Swiss Cheese model also emphasizes human error as a primary cause of accidents. One notable addition by Reason is the classification of errors into one of two categories: active errors and latent errors. Active errors are defined as errors "whose effects are felt almost immediately" while latent errors are defined as errors "whose adverse consequences may lie dormant within the system for a long time" [27]. However, critics argue that the distinction between active and latent is arbitrary and the boundary between "immediate" and "for a long time" is not always clear. Moreover, the classification is based on the eventual outcome of the action, which is usually not known at the time the action is performed and may not be useful in understanding why errors occur and how to prevent them.

2.1.4 Functional Resonance Accident Model (FRAM)

The Functional Resonance Accident Model (FRAM) was proposed by Erik Hollnagel in 2004 to capture emergent phenomena in complex nonlinear systems [36]. The model is derived from the idea of stochastic resonance [37], which describes the detection of a weak periodic signal superimposed on a stronger random noise component. The weak signal is such that it is normally undetectable due to the relative strength between signal and noise, and therefore the presence of the weak signal has very little affect on any downstream systems. However, in certain conditions, increasing the amount of random noise by a small amount can induce resonance in nonlinear components of the system, which counterintuitively allows downstream components to react to the weak signal resulting in better detection ability. Stochastic resonance was first described by Roberto Benzi in 1982 to explain the periodic behavior of the earth's ice ages [38, 39], and has since been studied in neurology [40-42], electrical engineering [43-45], ecological models [46], financial models [47], social systems [48], and game-theoretic strategies [49].

In Hollnagel's model the performance of any subsystem may be variable to some extent, which is seen as a weak modulated signal. The "signal" is normally weak enough to remain undetectable and within the tolerance of the system (no accident occurs). However, other subsystems also exhibit variable performance, and together the aggregated variability of other subsystems is modeled as a noise component. If the noise meets certain criteria, it can cause resonance in nonlinear components of the system that effectively increases sensitivity to the weak signal (i.e. component variability) beyond the tolerance of the system resulting in an accident. In other words, FRAM models how the combined performance variability of multiple nonlinear subsystems can create an environment in which otherwise acceptable and normal component deviations may lead to an accident.

Although adapted from the concept of stochastic resonance, Hollnagel uses the term *functional resonance* because the nature of the noise component in his model is fundamentally different. In stochastic resonance the noise component is purely random, but in FRAM the "noise" represents the performance variability of many subsystems and is not purely random. Hollnagel argues that the noise is determined by the functions and structure of the system and therefore the resulting resonance is more correctly labeled functional resonance.

FRAM has been applied to a few systems [50, 51], but it not been as popular as other accident models. Although stochastic resonance has been discovered and quantitatively established in many domains, Hollnagel's application assumes that the reaction of engineered subsystems to non-random and potentially undefined noise will mimic the reaction of well-defined nonlinear components to random noise. However, there has been little evidence to verify this claim. In addition, although the model is most applicable to accidents in which component variability is a factor, the model is not comprehensive and may be much less suitable for accidents with components that fail or components (e.g. software) that cause accidents by performing exactly as designed and do not exhibit variability.

2.1.5 System-Theoretic Accident Model and Processes (STAMP)

System-Theoretic Accident Model and Processes (STAMP) [10] was published by Nancy Leveson in 2002 [52, 53] to capture more types of accident causal factors including social and organizational structures, new kinds of human error, design and requirements flaws, and dysfunctional interactions among non-failed components. Rather than treating safety as a failure problem or simplifying accidents to a linear chain of events, STAMP treats safety as a control problem in which accidents arise from complex dynamic processes that may operate concurrently and interact to create unsafe situations. Accidents can then be prevented by identifying and enforcing constraints on component interactions. This model captures accidents due to component failure, but also explains increasingly common *component interaction accidents* that occur in complex systems without any component failures. For example, software can create unsafe situations by behaving exactly as instructed or operators and automated controllers can individually perform as intended but together they may create unexpected or dangerous conditions.

STAMP is based on systems theory and control theory. Complex systems are viewed as hierarchical structures with multiple levels; each level controls lower levels by imposing constraints on the level beneath it. Figure 4 shows a generic example hierarchical control structure. Control processes operate throughout the hierarchy whereby commands or *control actions* are issued from higher levels to lower levels and feedback is provided from lower levels to higher levels. Accidents arise from inadequate enforcement of safety constraints, for example due to missing or incorrect feedback, inadequate control actions, component failure, uncontrolled disturbances, or other flaws.

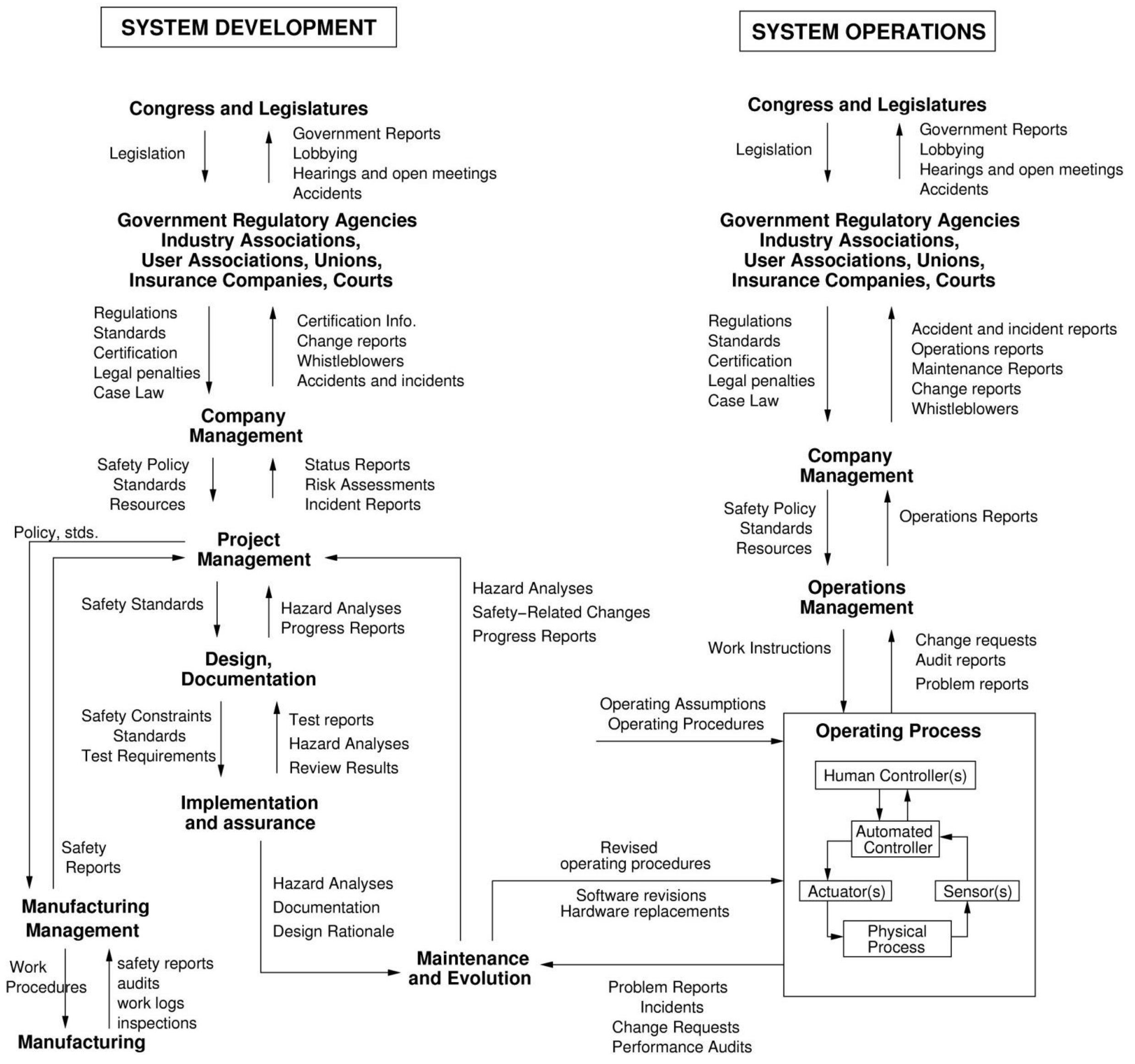


Figure 4: Generic example of a hierarchical control structure

STAMP defines four types of unsafe control actions that must be eliminated or controlled to prevent accidents:

1. A control action required for safety is not provided or is not followed
2. An unsafe control action is provided that leads to a hazard
3. A potentially safe control action is provided too late, too early, or out of sequence
4. A safe control action is stopped too soon or applied too long

One potential cause of a hazardous control action in STAMP is an inadequate process model used by human or automated controllers. A *process model* contains the controller's understanding of 1) the current state of the controlled process, 2) the desired state of the controlled process, and 3) the ways the process can change state. This model is used by the controller to determine what control actions are needed. In software, the process model is usually implemented in variables and embedded in the program algorithms. For humans, the process model is often called the "mental model" [52]. Software and human errors frequently result from incorrect process models; for example, the Mars Polar Lander software prematurely believed the spacecraft had landed and shut off the descent engines too early [54]. Accidents like this can occur when an incorrect or incomplete process model causes a controller to provide control actions that are hazardous. While process model flaws are not the only cause of accidents in STAMP, it is a major contributor.

STAMP has been successfully used in many domains including aerospace, defense, energy, chemical, healthcare, and transportation systems. STAMP is especially adept at capturing behavior in modern complex human- and software-intensive systems where component interaction accidents have become increasingly common and traditional chain of events models have proven inadequate. However, no formal structure has previously been defined for STAMP to permit the development of automated analysis methods based on the STAMP model.

2.2 Hazard analysis methods

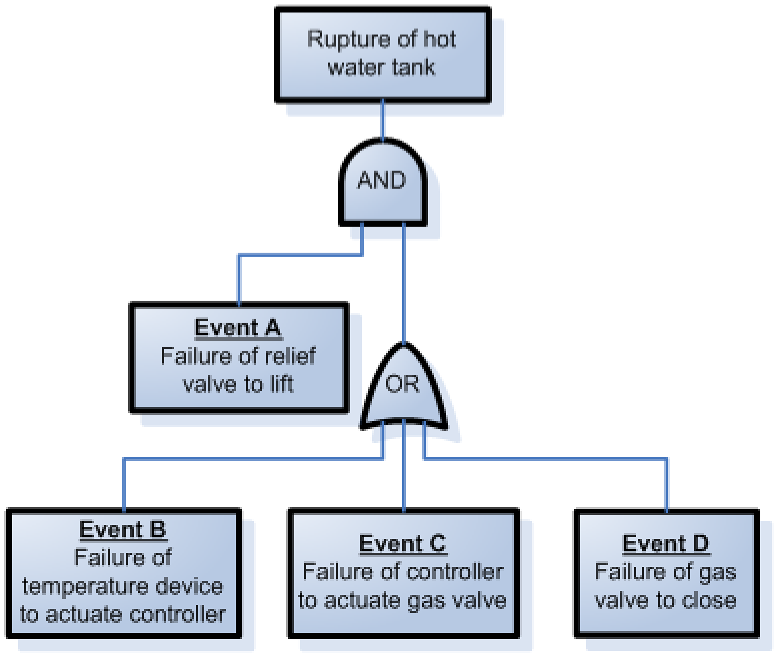
While accident models explain why accidents occur and what assumptions can be made, they do not specify what steps need to be taken to analyze a system or accident. There are two main classes of methods that use accident models: accident analysis methods and hazard analysis methods. Accident analysis methods describe how accident models can be applied to identify the causes of an accident that has already occurred. Hazard analysis methods describe how to use accident models to identify potential causes of a future accident in a system that already exists or is being designed. The following sections review current hazard analysis methods used today.

2.2.1 Failure-based methods

2.2.1.1 Fault Tree Analysis (FTA)

Fault Tree Analysis (FTA) was developed at Bell Laboratories in 1961 under a U.S. Air Force contract to analyze the Minuteman missile system [55]. FTA is based on the chain of events accident model, and was designed for component failure events. Electromechanical component failures were quite common at the time, but there was no method to analyze the many potential combinations of failures that could cause hazardous behavior like an accidental missile launch. Bell Labs developed FTA as a way to identify critical failure combinations, determine which combinations were most likely, and establish whether individual failure rates are sufficiently low. The analysis approach was first demonstrated on the Launch Control System of Minuteman I, and then extended by Boeing and AVCO to include components throughout the entire Minuteman II system [56]. Following its success on missile systems, FTA was adopted by organizations in many different industries and is now one of the most popular methods used during hazard analysis.

FTA begins with an undesirable event, such as an inadvertent missile launch or aircraft engine failure, and proceeds in a top-down fashion to identify the causes of the undesirable event in progressive levels of detail. The result is documented in a tree structure, where high-level undesirable events or faults are caused by combinations of lower-level component failures. A *failure* is an event in which a component does not operate in accordance with its specification, for example if a relay fails to close properly when a voltage is impressed across its terminals. A *fault* event describes component behavior that results from a failure and causes an unsatisfactory state, such as a relay closing at the wrong time due to the improper functioning of an upstream component [1, 5]. Events at the top of the tree describe faults while the lowest-level events, called primary events, describe failures. Figure 5 shows an example fault tree from the original Bell Laboratory report.



Rupture of hot water tank
AND
Event A
Failure of relief valve to lift
OR
Event B
Failure of temperature device to actuate controller
Event C
Failure of controller to actuate gas valve
Event D
Failure of gas valve to close

Figure 5: Example fault tree from the original Bell Laboratory study [55]

Events at each level are decomposed using either OR logic or AND logic into more detailed events. AND logic is used to indicate that an event will occur only if all events in the immediately lower level occur. OR logic indicates that the event will occur if any events in the immediately lower level occurs.

In 1981, Vesely further refined and standardized FTA with a number of rules, symbols, and rationale for connecting nodes in a fault tree [1]. For example, he argued that causality passes through AND but never through OR gates; inputs to OR gates should always be "identical to the output but more specifically defined as to the cause" [1]. He also argued that fault trees should not be constructed with direct connections between gates, as in Figure 5, but instead the intermediate events should always be identified and labeled, as in Figure 6. Vesely's work became very popular, and most applications of fault trees today adopt the conventions he endorsed [57-59].

When the fault tree is complete, it can be analyzed to identify combinations of component failures or *cut sets* sufficient to cause the top-level undesirable event. For example, one cut set for the fault tree in Figure 5 consists of event A together with event B; another cut set consists of event A together with C and D. The former cut set also a *minimal cut set* because it cannot be further reduced into a smaller cut set. Minimal cut sets from a fault tree can be used to help prioritize the importance of component failures and focus engineering efforts. For example, failures that appear in every minimal cut set—such as event A in Figure 5—might warrant a higher ranking than other failures [60].

If the component failure rates or probabilities of individual failures are known then a quantitative analysis may be performed to calculate the likelihood of the top-level event. In a quantitative analysis, the failure events are typically assumed to occur independently, which greatly simplifies the analysis and does not require the measurement of complex dependent failure rates. Although the independence assumption is often made for physical devices, it may not be valid if the failure rates are substantially affected by changes in independent variables such as temperature, vibration, mechanical stresses, etc. For example, the two O-rings involved in the loss of the Challenger shuttle were originally believed to be independent, but it was later discovered that certain failures of the first O-ring would inevitably lead to the failure of the second O-ring [61].

2.2.1.1.1 FTA Evaluation

FTA is a powerful top-town method of analyzing combinations of failures that can cause an undesirable event. The method was designed for engineered components with well-known failure modes and effects and has been very successful when applied to physical systems where most accidents are due to component failures. In fact, techniques like FTA have been so effective at preventing component failure accidents that a new type of accident—component interaction accidents—are becoming much more predominant in modern complex systems. More and more accidents are being caused by problems such as flawed requirements, design errors, and unsafe interactions among components that have not failed. However, these causes are much more difficult to capture in a fault tree.

Software errors are notoriously difficult to capture in a fault tree because software does not fail; software often causes accidents by performing exactly as instructed. Unlike physical components with a small number of well-known failure modes that can be added to a fault tree, software can misbehave in a practically infinite number of unpredictable ways. Some attempts have been made to include software in fault trees by adding a single box labeled “software error” or “computer failure” as in Figure 6, but doing so doesn’t provide software engineers and developers with the information they need to ensure the software will be safe. Other attempts add very specific software behaviors to the fault tree [62, 63], but these are either incomplete or they quickly become incredibly complex and inefficient when applied to modern complex software. In practice, software errors are often ignored completely in fault trees [5, 64], as well as design errors and requirement flaws for the same reasons. Human behavior is also extremely difficult to capture in a fault tree because humans are adaptive creatures that can learn to react in new ways and can respond to unexpected situations. In fact, a major reason why humans are still chosen to operate safety-critical systems like aircraft and nuclear reactors is to handle exactly those unexpected situations that engineers might not have anticipated.

Because FTA begins with an undesirable event, some other method must be used first to identify the set of undesirable events to be analyzed with FTA. Another issue is identifying the lower-level events. Although some limited guidance is provided in terms of when certain logical operators can be used and how faults can be combined, much less guidance is provided for identifying the actual faults and failures in the tree. As a result, many completed fault trees are later found to omit important events. For example, the fault tree in Figure 6 was produced in 1983 for an aircraft collision avoidance system but omits the possibility that a conflict alert is displayed and the controller does not observe it. Similarly, the fault tree in Figure 7 was produced in 2008 for new NextGen procedures but omits the possibility that Air Traffic Control does not check the Mach differential.

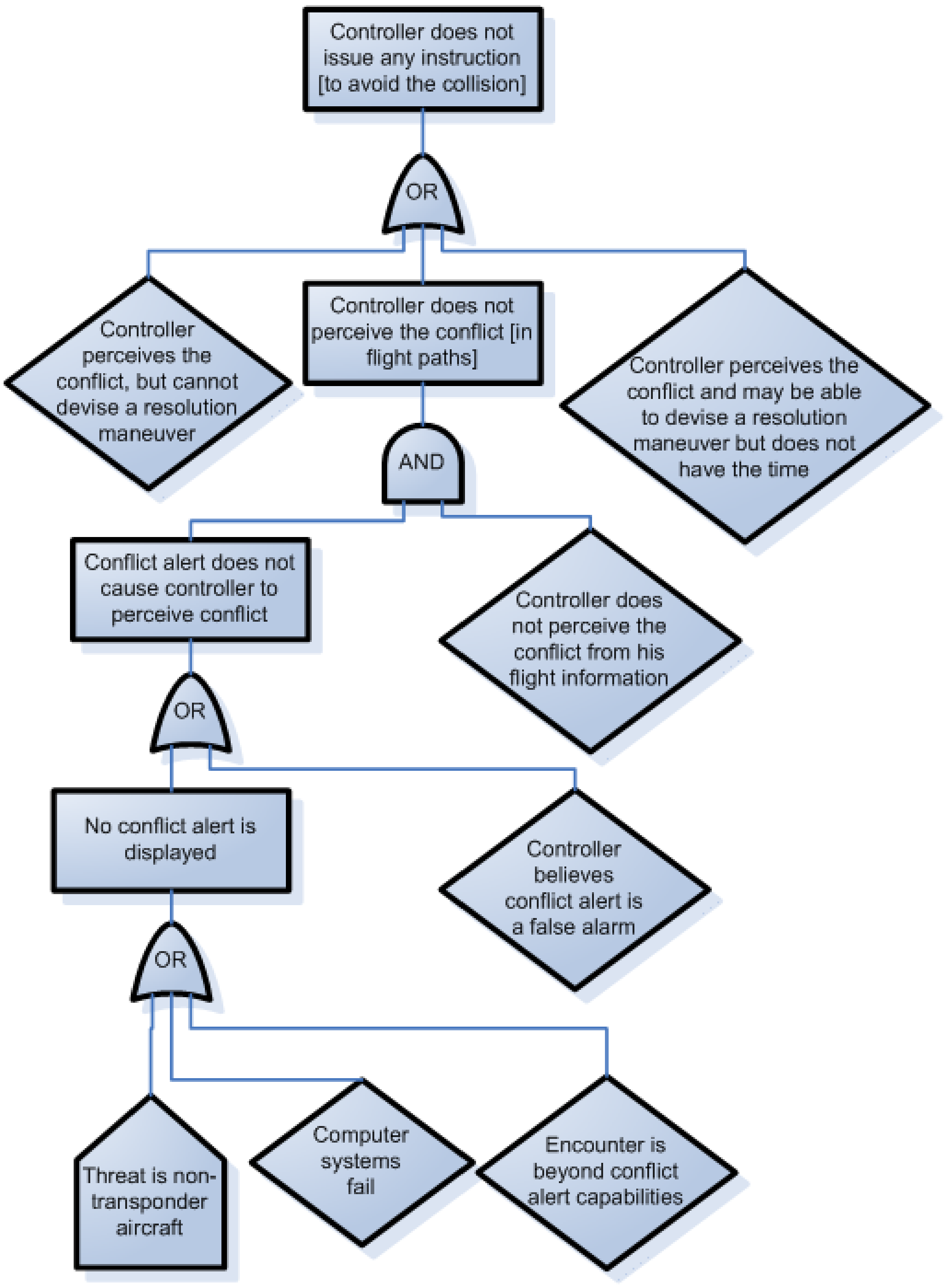

Controller does not issue any instruction [to avoid the collision]
OR
Controller perceives the conflict, but cannot devise a resolution maneuver
Controller does not perceive the conflict [in flight paths]
Controller perceives the conflict and may be able to devise a resolution maneuver but does not have the time
AND
Conflict alert does not cause controller to perceive conflict
Controller does not perceive the conflict from his flight information
OR
No conflict alert is displayed
Controller believes conflict alert is a false alarm
OR
Threat is non-transponder aircraft
Computer systems fail
Encounter is beyond conflict alert capabilities

Figure 6: A partial fault tree for an aircraft collision avoidance system [65]

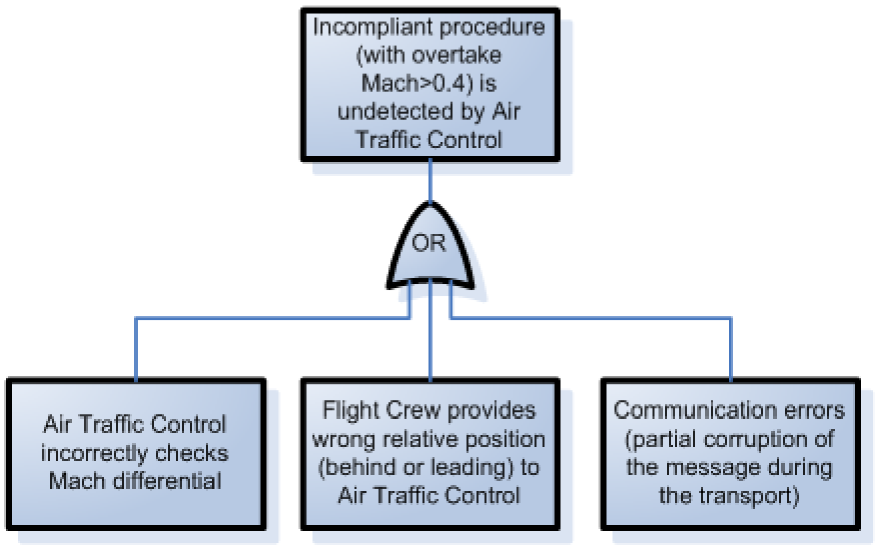

Incompliant procedure (with overtake Mach>0.4) is undetected by Air Traffic Control
OR
Air Traffic Control incorrectly checks Mach differential
Flight Crew provides wrong relative position (behind or leading) to Air Traffic Control
Communication errors (partial corruption of the message during the transport)

Figure 7: A partial fault tree for proposed airspace procedures [66]

At each level, FTA users must seek out additional information and identify lower level causes, but there is no systematic method for doing so. The analysis itself—i.e. finding causes and linking them together—is performed mentally based on one’s own experience and knowledge; the fault tree simply documents the output of the analysis. The analysis must also be based on some existing system model, but because FTA does not include any standard system model a mental model is typically used instead. For these reasons, there is no way to check or verify that all the causes have been identified at any given point or that all users are operating on the same understanding and assumptions of the system. Meanwhile, some of the most important contributors to accidents in complex systems today involve subtle behavior that was never anticipated or not included in the developers’ mental model. Because FTA relies on an existing model of the system it is also less useful for driving critical decisions during early stages of development when a detailed system model does not yet exist.

Another disadvantage is the lack of a stopping rule when performing FTA. Failure and fault events can almost always be decomposed further, and a subjective assessment must always be made about when to stop. For example, the diamond shapes in Figure 6 indicate events that are not further decomposed in the analysis. The lowest-level boxes Figure 7 were not decomposed further either. In practice, decomposition often stops when the causes are no longer obvious or become too complex. However, the subtle or complex factors are often the most important ones to examine, especially for software- and human-intensive systems. For example, the event “controller believes conflict alert is a false alarm" cannot be addressed without understanding why that behavior may occur (e.g. the system may have generated too many false alarms in the past). These explanations, which may not be events or even faults, are not easily included in FTA. As others have noted, FTA often finds only what is already intuitively obvious. [5, 67]

Although quantitative FTA was originally developed to analyze combinations of electromechanical device failures, various attempts have been made since its original inception in the 1960s to extend quantitative FTA to other types of components. Boeing used simulators to attempt to identify and quantify potential pilot errors for use in a fault tree as early as 1968, but noted that the human system was far too complex for an accurate assessment and that developing numerical probabilistic values was slow and painful process [68]. A number of improvements have been made since then, but the same limitations are still being observed:

> *Operators do not get simulator sickness as do pilots, they do not have to make billion dollar tradeoffs which they might in an actual severe accident, and the simulators themselves can only simulate 'standard, textbook scenarios.' .... 'Hot cognition', decision making under fire and uncertainty, is just not elicited in a simulator. [69]*

> *The most serious problem ... continues to be the same problem that was recognized in the early 1960s among HRA practitioners—the scarcity of data on human performance that are useful for quantitative predictions of human behavior in complex systems. ... Except for use of highly fallible expert judgment, the training simulator is the only practical method for collecting data on human errors that have very low probabilities of occurrence (1E-4 or smaller) or which are associated with operating conditions that would occur with a very low frequency. Yet the simulator is not the real world. How can raw data from training simulators be modified to reflect real-world performance? This is not a small problem. [70]*

Expert judgments have also been utilized as a way to identify and quantify probabilities of operator errors in a fault tree [66, 71]. In practice, this approach is typically used when there is little or no objective data available for the quantity of interest and critics argue that it is therefore not possible to validate (or disprove) the expert estimates that are used [72]. Expert estimates are also subject to a number of cognitive biases: estimates are almost always overconfident, usually over- or under-estimate the quantity of interest, and vary significantly between experts [73, 74]. Although some methods have been proposed to reduce systematic biases, critics argue that such approaches only improve inter-judge reliability and do not necessarily validate the estimates themselves [72, 73].

2.2.1.2 Event Tree Analysis (ETA)

Event Tree Analysis (ETA) was developed during the WASH-1400 nuclear power plant safety study in 1974 [75, 76]. A comprehensive fault tree analysis was originally attempted for this task, but was deemed too large and cumbersome to be practical [5, 75]. Event trees were originally conceived as a way to condense the analysis by defining potential accident paths so that each failure in the path can be further analyzed using a fault tree. Although event trees were originally designed to be combined with fault trees as part of an overall Probabilistic Risk Assessment, Event Tree Analysis (ETA) has also been introduced as a separate method in its own right [75, 77-79]. Like FTA, ETA is based on the chain of events accident model.

A simplified event tree for a nuclear reactor is shown in Figure 8. The first step is to identify an initiating failure event such as a ruptured pipeline or loss of power. Next, the set of barriers or protective functions intended to prevent the initiating event from leading to an accident are listed in the anticipated sequence of operation. Finally, a logical tree is constructed by tracing forward in time from the initiating event and inserting a binary branch at each barrier to represent the possible success or failure of that barrier.

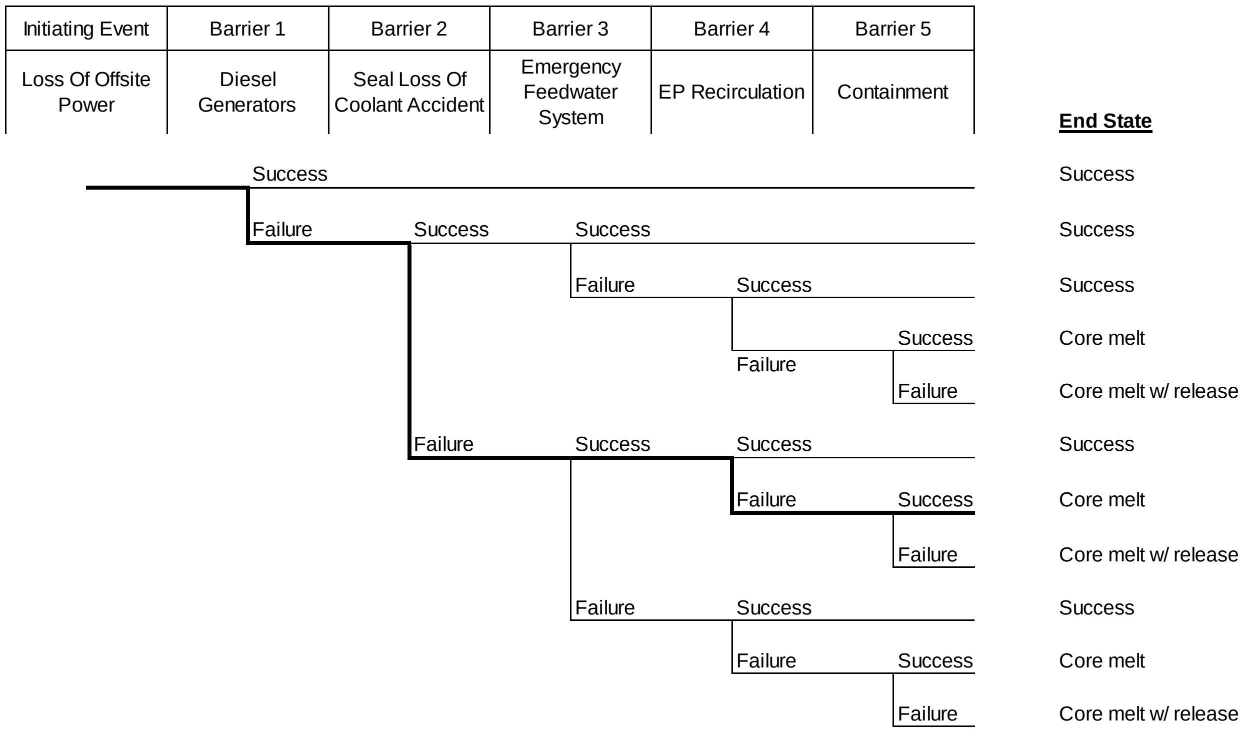


Figure 8: Simplified event tree for a nuclear reactor adapted from [80]

Like fault trees, the structure of an event tree lends itself well to a quantitative analysis if the probabilities of each barrier's success or failure are known. In practice each barrier is often assumed to operate independently, which allows computing the probability of each end state (conditioned on the initiating event) by simply multiplying the probabilities of success or failure along each path to the end state. The end state probabilities can also be calculated if the barriers operate dependently and the probability of each barrier's success or failure (conditioned on the success or failure of the previous barrier) is known.

2.2.1.2.1 ETA Evaluation

Event Trees Analysis is a useful way to examine the anticipated effects of physical protection systems when the probabilities of failure are known, but like any method there are several limitations. Event trees must start with an initiating event, but do not provide a way to systematically identify the initiating events or to be sure that all relevant initiating events are included. Some other method must be used to identify the initiating events that need to be considered. In addition, because the analysis starts by assuming the initiating event has occurred, the method focuses on functions to mitigate its consequences; preventative measures to avoid the initiating event are not considered.

When human behavior is included in an event tree, human actions are reduced to a binary decision that is equated to a success or failure in the tree. Critics argue that this simplification can mask the wide range of behaviors possible at any given moment and removes critical context that explains *why* a person would choose a given action [3]. Human behavior is intimately connected to and influenced by the context in which it occurs: the information available, goals, past experiences, beliefs about the current system state, interpretation of various observations, etc. By removing the context, preventative measures to ensure safe behavior are easily overlooked.

Various extensions have been developed to better capture human behavior in event trees including Dynamic Event Trees that model stochastic variations in operating crew states [81], time-dependent event trees that include certain operator interventions [82], and Fuzzy Event Tree Analysis (FETA) that employs fuzzy logic to integrate human errors into event trees [83]. However, these extensions all employ the same basic chain of events accident model and inherit the same weaknesses including oversimplification of human behavior as binary decisions (e.g. success/failure), assumptions of a pre-defined sequence of barriers or process events, and emphasis on deviation from specified procedures rather than evaluating how the procedures may be flawed or inadequate. With a few exceptions (e.g. ATHEANA [84]), these extensions also assume human behavior is random and tend to focus on quantification and assessment of operator errors rather than explaining the underlying reasons or providing engineering insight to prevent errors.

Although the barriers in an event tree are often assumed to operate independently of each other, in practice they may not be truly independent. For example, in the recent Fukushima accident the loss of offsite power and the loss of the diesel generators were not independent events as Figure 8 suggests; they were both caused by the same factors. In general, the behavior of multiple barriers can be heavily dependent on the same set of factors, especially if human behavior is involved. For example, in the infamous Three Mile Island (TMI) accident, the operators were unaware of steam forming in the core and they manually disabled the primary loop pumps and the emergency core cooling pumps. Clearly, the failure of these barriers to operate was not independent.

Critics also argue that design errors and requirements flaws are critical factors that cannot be analyzed with an event tree [5, 85]. In the example above, an important reason TMI operators did not initially believe coolant was being lost is that an indicator lamp erroneously suggested that a relief valve was closed and a water level indicator erroneously suggested the water level was sufficient. Both of these instruments satisfied their individual requirements and, in fact, operated exactly as designed, but the design and requirements were flawed. The indicator lamp was designed such that it would not reflect the actual state of the valve when the valve became stuck, and the water level indicator was designed such that it overestimated the amount of water present when steam became trapped in the reactor core. Design and requirements flaws such as these are not amenable to analysis using an event tree and are easily overlooked.

Higher-level systemic causes such as organizational and managerial issues are also omitted from an event tree [5, 86]. For example, poor management, ineffective communication, misplaced regulatory priorities, and complacent attitudes were important contributors at TMI [87] and simultaneously affected the efficacy of multiple barriers, but these aspects are all omitted in event trees. Event trees also omit non-linear or feedback relationships that can contribute an accident, such as two or more processes that mutually affect each other. For example, the operators at TMI initially believed that coolant was not being lost because their interactions with the system reinforced this belief. Processes operating at a much higher level are also important; for example, low accident rates can give rise to complacency and increased pressures to reduce budget and oversight, which in turn leads to higher accident rates [88]. These aspects are overlooked by event trees.

Note that many of these omitted factors are also missing in FTA; combining event trees and fault trees may improve some aspects of the analysis but it does not address these critical factors that are missing from both techniques. This is discussed in more detail in section 2.2.1.4.

2.2.1.3 FMEA and FMECA

Failure Modes and Effect Analysis (FMEA) and its cousin Failure Modes Effects and Criticality Analysis (FMECA) were developed by reliability engineers to systemically evaluate the effect of individual component failures on system performance [77]. Both approaches are based on the chain of events accident model. They were first introduced as a procedure for weapons systems in 1949 [89], and in 1955 a similar procedure was introduced by the U.S. Navy Bureau of Aeronautics [90]. In the 1960s these methods were refined and adopted by the aerospace industry, and they began to be applied on a number of NASA programs including Apollo and Voyager [91, 92]. By the 1970s they were being used in civil aviation, the automotive industry, and even offshore petroleum exploration [91, 93-95]. Today FMEA and FMECA are used across a broad array of fields including food, drug, and cosmetic industries [96].

Due to its popularity, FMEA has been implemented in a number of different ways but generally follows the same bottom-up approach. First, the various components in the system are identified. Next, the failure modes—defined as mechanisms by which a component may fail to achieve its designed function—are identified [97, 98]. For each failure mode, the potential causes and effects on the system are investigated. FMECA follows the same basic process, but in addition assigns a criticality to each failure mode by examining the severity and probability of each identified effect.

Table 1 below shows an example FMECA worksheet that summarizes the analysis.

Table 1: Example FMECA worksheet adapted from [99]

Component	**Failure Mode**	**Cause**	**Effect**	**Severity**	**Probability of Occurrence**	**Criticality**
Water Tank	Leak	Corrosion	Lost water	Catastrophic	0.0001	High
Valve	Stuck closed	Dirt, corrosion	No water	Catastrophic	0.00012	Very High
	Stuck open	Corrosion, power	False trip	Marginal	0.0002	Low

Note that FMEA/FMECA can be applied to either physical or functional models of the system, although in practice physical and functional applications can overlap significantly and are not always distinct. For example, if applied to a physical model then the analysis of failure modes, effects, and severity are still identified with respect to the designed function of each component, and if applied to a functional model then the causes and failures may still be based on the physical implementation of the components. [97, 98, 100]

2.2.1.3.1 FMEA and FMECA Evaluation

FMEA and FMECA are useful methods for analyzing the reliability of physical system components and identifying single points of failure that may lead to an accident. However, there are a number of limitations especially when applied to other goals or other types of components. Because these methods start by identifying low-level failures to consider, the resulting scenarios that are analyzed include both hazardous and non-hazardous scenarios triggered by a failure. Both types of scenarios are analyzed in similar levels of detail. If the goal is safety-related, then the effort spent analyzing non-hazardous failures may require significant time and effort without adding value to the analysis.

More importantly, the set of scenarios triggered by a failure does not include all unsafe scenarios, as illustrated in Figure 9. For example, if the system requirements are flawed then the emergent behavior of the system may be unsafe even though all components operate exactly as designed and required. Filtering out all scenarios that do not begin with a failure effectively excludes these types of hazardous scenarios.

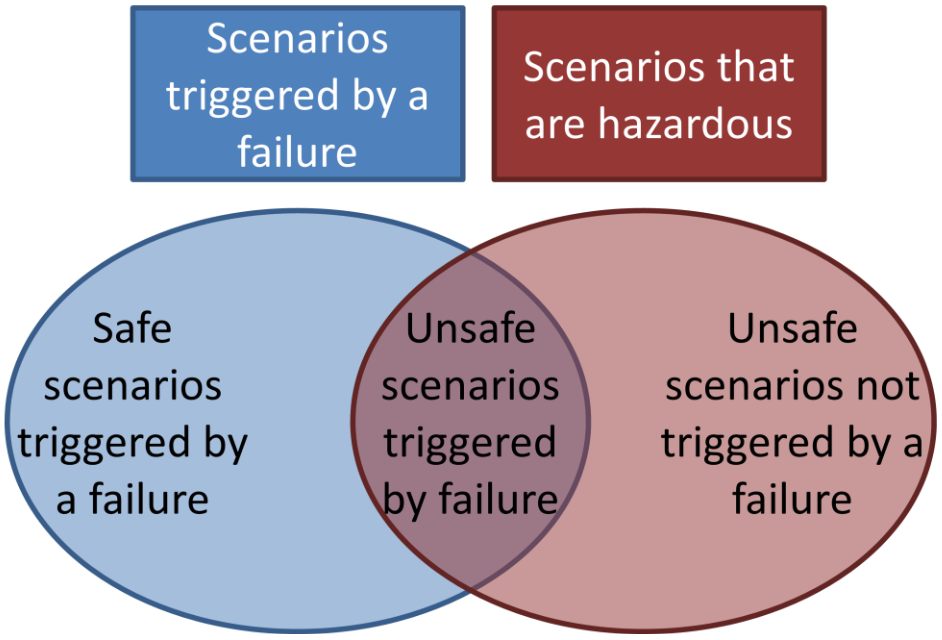

Scenarios triggered by a failure
Scenarios that are hazardous
Safe scenarios triggered by a failure
Unsafe scenarios triggered by failure
Unsafe scenarios not triggered by a failure

Figure 9: A Venn diagram of failure scenarios and unsafe scenarios

Like other methods, FMEA/FMECA assume a linear progression of events and do not capture non-linear and feedback relationships. Like ETA, FMEA/FMECA only consider scenarios initiated by a single failure and omit scenarios that result only from a combination of several failures. By focusing only on single failures, only a subset of all failure scenarios (the left circle in Figure 9) are analyzed.

2.2.1.4 General evaluation of failure-based methods

One of the most important limitations of failure-based methods is that by definition they omit entire classes of factors that lead to accidents. Failure-based methods are generally designed to capture the propagation of component failures in a system that causes an undesired event. However, many causes of accidents do not involve any component failure. With today's systems becoming increasingly complex, more and more accidents are occurring not due to component failures but instead due to critical design errors or requirements flaws. In addition, socio-technical systems tend to exhibit dynamic non-linear behavior that is difficult or impossible to capture with a technique designed for a linear propagation of faults. Continuously adaptive behavior, goal-seeking behavior, local optimization with global deterioration, goal erosion, mutually reinforcing relationships with exponential growth or collapse, and budgetary/financial pressures are just a few factors that can have a critical impact on the safety of a system. However these are dynamic processes, not independent failure events, and are not explained with a one-way linear fault propagation structure.

Similarly, human error in a failure-based method is treated in exactly the same way as a hardware failure—as a deviation from a specified behavior or procedure. However, like software, the number of potential ways a human can deviate is virtually infinite. Even if all noncompliant behaviors could be listed, it is not sufficient to just identify them; in order to prevent a behavior it is necessary to understand *why* a person might behave that way. In other words, it requires first understanding the conditions under which unsafe decisions might make sense to a person at the time and then modifying or adding requirements to make the correct decisions obvious. Unfortunately, framing human error as a failure requires oversimplifying human behavior as a binary decision between right and wrong—a determination that is often only clear in hindsight and does not reflect the perspective of the person at the time. In fact, this over-simplification can obscure the underlying reasons for the behavior, including many important causal factors that are difficult or impossible to model in a failure-based method such as:

- Correct human behavior that is not defined for certain situations
- Specified human behavior that is known by operators but thought to be incorrect
- Procedures that conflict with each other, or it is not obvious which procedure applies
- Information necessary to carry out a procedure is not available or is incorrect
- The person has multiple responsibilities or goals that may conflict
- Past experiences and current knowledge conflict with a procedure
- Procedures are not clear or misunderstood
- Procedures are known but responsibility for the procedures is unclear or misunderstood
- Procedures are known and followed, but they are unsafe

Consider an example[2]: In the 2010 Deepwater Horizon oil spill, a critical factor was that workers reported a successful negative pressure test when in reality oil had already begun seeping into the well. [101, 102] The workers did not know that earlier tests had clogged a pipe that rendered a key instrument reading invalid. Note that in this case the behavior was compliant—the workers followed procedures but the procedures were inadequate and unsafe. The behavior was not a "failure event" because nothing failed – the flaw existed from the beginning in the form of inadequate procedures and feedback for the crew. A failure-based method could help focus engineering efforts on preventing the pipe from getting clogged or perhaps preventing workers from deviating from procedures, but would not help address the flawed requirements, inadequate procedures, and inadequate feedback loops that existed. For example, a potential solution that adds equipment to detect a clogged pipe and adjusts worker procedures to utilize this information would be masked by a failure-based method that focuses only on preventing clogs and enforcing worker compliance with existing procedures.

Most failure-based methods were originally designed and developed to handle simple electromechanical components. Numerous attempts have been made to extend these methods to other components like software, but software is fundamentally different in the sense that it does not fail like hardware does. Unlike hardware, software always behaves exactly the way it was

---

[2] This example is necessarily an oversimplification of the complex events that unfolded on the Deepwater Horizon, but it is nevertheless a useful illustration for the point being made.

programmed (and therein lies the problem). If software exhibits unsafe behavior, it is because that unsafe behavior was programmed from the beginning—not because the software “wore out” or “broke” over time. Because of this fundamentally different problem and the virtually infinite number of ways software can be programmed incorrectly, it is very difficult to capture software-related causes in a failure-based method. In practice, failure-based methods often ignore software errors completely [5, 64] or include them under a generic label of “software failure” that is not decomposed further. However, simply stating that the software could cause an undesirable event offers little assistance to the system and software developers who need to make the software safe.

Failure-based methods are often applied quantitatively to consider the probability of certain failures and outcomes. Assuming independence between different failure events is very common and can significantly simplify the analysis, but this assumption is often made incorrectly. For example, the primary and backup O-rings on the Challenger shuttle were originally believed to be independent and redundant[3] [61]. Unfortunately, they weren’t truly independent because low temperature and mechanical pressures affected both O-rings and contributed to their simultaneous failure in the famous 1986 accident. While assuming independence between failure events may simplify the probabilistic calculations, doing so has often resulted in overconfident probabilities for hazardous events.

Accurately quantifying probabilities for software errors is difficult or impossible. Even if all potential software errors could be listed for a simple system, predicting their probability of occurrence is not feasible. An error is either known to exist or not known to exist. Critics argue that if an error is ever known, it’s far more effective to simply fix it than to add events to a fault-based model or guess a probability of occurrence [5].

Although software errors are important, the majority of software-related accidents can be traced to flawed requirements rather than a problem with the software implementation itself [103, 104].

[3] The SRB O-ring joint criticality status was originally classified as C 1R (redundant). Marshall eventually proposed changing the status to C 1 (non-redundant), but Thiokol engineers still disagreed with the change and argued that it should remain C 1R. Although the status was later officially changed to C 1 in some databases, this issue remained an important factor in the resulting accident. [61]

Clearly in any system—whether dealing with software, hardware, or even human components—safety is dependent on having correct and safe requirements. However, there is no empirical data for quantifying the probability that a requirement is flawed. Like software instructions, if a requirement is known to be flawed, it is far more effective to fix it than to guess the probability that it is wrong.

Requirements provided in the form of procedures for human operators are also critical for ensuring safety. For example, inadequate procedures played an important role in the Three Mile Island partial nuclear meltdown in 1979. Many operating and emergency procedures contained substantive errors, typographical errors, imprecise or sloppy terminology, and violated the nuclear reactor's specifications [5]. As with requirements, there is typically no data to support a probabilistic estimate of a flawed procedure before an accident. Even if such an estimate had been produced for Three Mile Island before 1979, it most likely would have been incorrect; before the accident the procedures were thought to be safe. Only afterward in hindsight were the flaws discovered.

Given the lack of a probabilistic estimate for these problems, it's easy to focus on creating methods to produce such estimates. However, it's important to recognize that the problem is much larger than just the lack of a quantitative probability. Suppose such an estimate did exist for requirements. What would the number mean? Any value other than 0% or 100% just indicates a lack of knowledge—i.e. it is not known whether a given requirement is flawed or safe, and the engineering task is therefore incomplete. The core issue is therefore not a difficulty quantifying existing knowledge; the core issue is obtaining the right knowledge in the first place. Addressing these problems will require better methods for finding flaws and creating safe requirements, not methods that estimate what is already known.

When human behavior is included in a quantitative failure-based analysis, the analysis typically assumes that the behavior is random with a given probability. However, human behavior is not random—it is heavily influenced by the context in which is appears and only appears random if we ignore the most important factors that explain it. For example, in the 2005 Texas City explosion a critical factor is that operators did not follow standard operating procedures to release hydrocarbons safely via the 3-pound venting system [105]. Instead, they bypassed the venting system and released flammable hydrocarbons through a blowdown stack into open air, contributing to the accident. In the absence of any knowledge about the system it might appear that these operators "flipped a coin" and randomly decided whether to follow the procedure, but this is far from true. The decision was a direct result of influence from supervisory personnel who advocated the bypass because it significantly shortened the startup time and had been used successfully many times in the past [105, 106]. With this additional knowledge, the operators' behavior does not appear random at all—it was both predictable and preventable given the context in which it occurred. While quantitative failure-based methods tend to isolate behavior from context by emphasizing human actions as random events, a better understanding of the context can often lead to a more accurate perception and much more effective solutions.

Although human error is often only used to refer to behavior during the operation of a system, it also applies to the development of a system. For example, software errors and flawed requirements are really just forms of human error. In fact, even hardware failures can be traced back to human decisions regarding the design and construction of the component, the selection of the component for a specific purpose in an assumed operating environment, the design of the system that interfaces with the component, and the inclusion of any protective measures that detect and handle (or don't) the potential failure of the component. Therefore it is not surprising that the same issues that plague software errors and flawed requirements appear again for human behavior in general.

2.2.2 Systems-based Hazard Analysis Methods

2.2.2.1 Hazard and Operability Analysis (HAZOP)

Hazards and Operability Analysis (HAZOP) was developed in 1964 by Imperial Chemical Industries (ICI) in England [85], although the method was not published until 1974 [107]. HAZOP was developed to help multidisciplinary teams identify ways chemical processes can lead to accidents. The analysis starts with a firm design [108], including a full description of design intentions, and proceeds to identify parameters in various parts of the system as shown in Table 2. For each parameter, a set of guidewords are applied to identify how the system may deviate from the design intention.

Table 2: HAZOP parameters and guidewords (adapted from [108])

Parameter	**Guidewords**
Flow	• None • More of • Less of • Reverse • Elsewhere • As well as
Temperature	• Higher • Lower
Pressure	• Higher • Lower • Reverse
Level	• Higher • Lower • None
Mixing	• Less • More • None

Once the potential deviations have been identified, they are evaluated to determine whether the consequences are hazardous and, if so, to identify possible causes of the deviation. A flow diagram of the HAZOP process is shown in Figure 10.

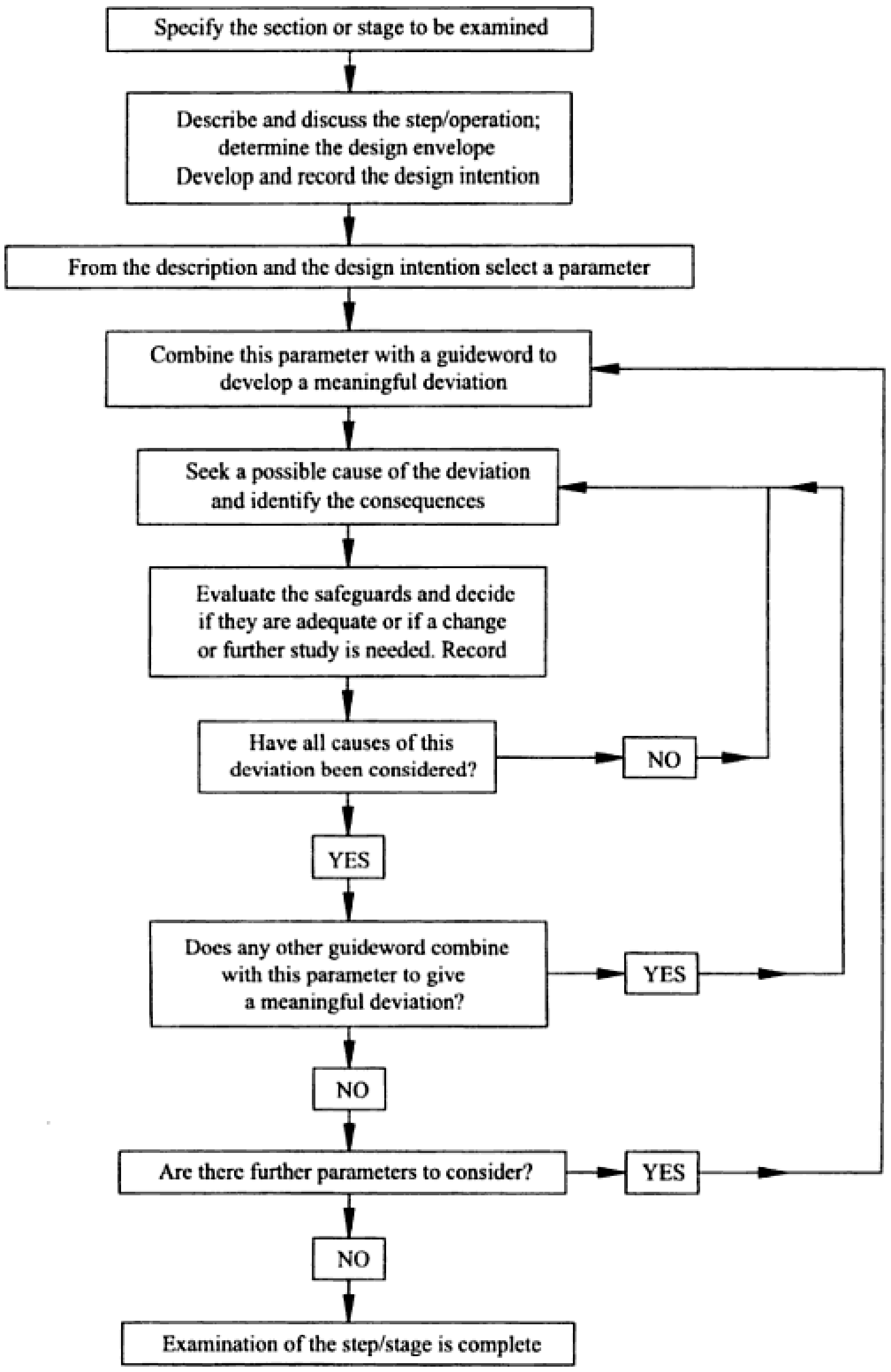

Specify the section or stage to be examined
Describe and discuss the step/operation; determine the design envelope Develop and record the design intention
From the description and the design intention select a parameter
Combine this parameter with a guideword to develop a meaningful deviation
Seek a possible cause of the deviation and identify the consequences
Evaluate the safeguards and decide if they are adequate or if a change or further study is needed. Record
Have all causes of this deviation been considered?
NO
YES
Does any other guideword combine with this parameter to give a meaningful deviation?
YES
NO
Are there further parameters to consider?
YES
NO
Examination of the step/stage is complete

Figure 10: HAZOP Flow Diagram (from [108])

2.2.2.1.1 HAZOP Evaluation

A notable advantage of HAZOP is its simplicity [5, 109]. The emphasis on deviations from intended behavior is also quite powerful and often captures accident causes beyond component failures. However, the method requires a fairly detailed system model to identify parameters and apply guidewords and is most often applied after detailed development phases when many important design decisions have already been made. This limits the number of potential solutions to minor changes like patches or protection systems added to an existing design.

Although HAZOP is most popular in the process industries, it has been applied in other domains and several extensions have been developed to accommodate human and software behavior [110]. A number of Human HAZOP variants have been developed to analyze human deviations from procedures [111-114]. Using similar guidewords, human tasks are identified and analyzed to find potential procedural deviations that may cause accidents. However, while these approaches provide ways to describe and classify human errors, they do not explain non-trivial errors and do not examine underlying causes of human behavior such as operator mental models. Human HAZOP methods have also been criticized for only focusing on errors during operation without considering broader organizational decisions or design flaws that can make human deviations from procedures more likely or inevitable [115].

Software Hazard Analysis and Resolution in Design (SHARD) is an approach inspired by HAZOP to identify potentially hazardous software behavior [116]. Like HAZOP, SHARD begins with a proposed design. Data flows in the software system are identified and a set of guidewords including *omission, commission, early,* and *late* are applied to each data flow. The result is evaluated in terms of possible effects and causes, and documented in a worksheet much like a FMEA. A number of Computer HAZOP (CHAZOP) variants have also been defined for applying HAZOP to computer systems [117, 118]. For example, guidewords such as *early, late, before,* and *after* may be applied to attributes such as *data flow, control flow, data rate, event, response time*, etc. Many similar CHAZOP variants have also been described in the literature [119-122].

Software HAZOP methods have been criticized for ambiguity, incompleteness, nonsensicality, inefficiency, and redundancy [123]. They have also been criticized for being time and labor intensive [117], and for the ad-hoc schemes that are used to derive the guidewords [118]. Although

HAZOP is a powerful technique, bottom-up approaches that start at the software level can be inefficient because both hazardous and nonhazardous data flows must be analyzed, and the analysis can be overwhelming when performed on complex software systems with large quantities and varieties of data flows. In addition, isolating individual attributes or flows can mask more complex problems that arise only when multiple attributes or flows interact in complex ways. In many cases, data or other information flows may not be the right units of analysis—for example, in process-oriented control systems with very little flow of information but with complex control algorithms and coordination schemes. Another important issue is that HAZOP relies on the user's understanding of the software behavior, interactions, and effects on other systems. While the physical pipe-and-process diagrams that HAZOP was originally created to analyze were relatively straightforward and exhibited well-understood behavior, today's complex and integrated software systems are fundamentally different and often cause problems precisely because they behave in ways that were unexpected or never anticipated [124].

2.2.2.2 Functional Resonance Accident Model (FRAM) Analysis

In 2004, Erik Hollnagel proposed an analysis method based on the Functional Resonance Accident Model (FRAM) [36, 50]. To apply FRAM, the functional entities of a system must first be defined along with their interdependencies and couplings. For this purpose, Hollnagel proposes a hexagonal functional representation as shown in Figure 11.

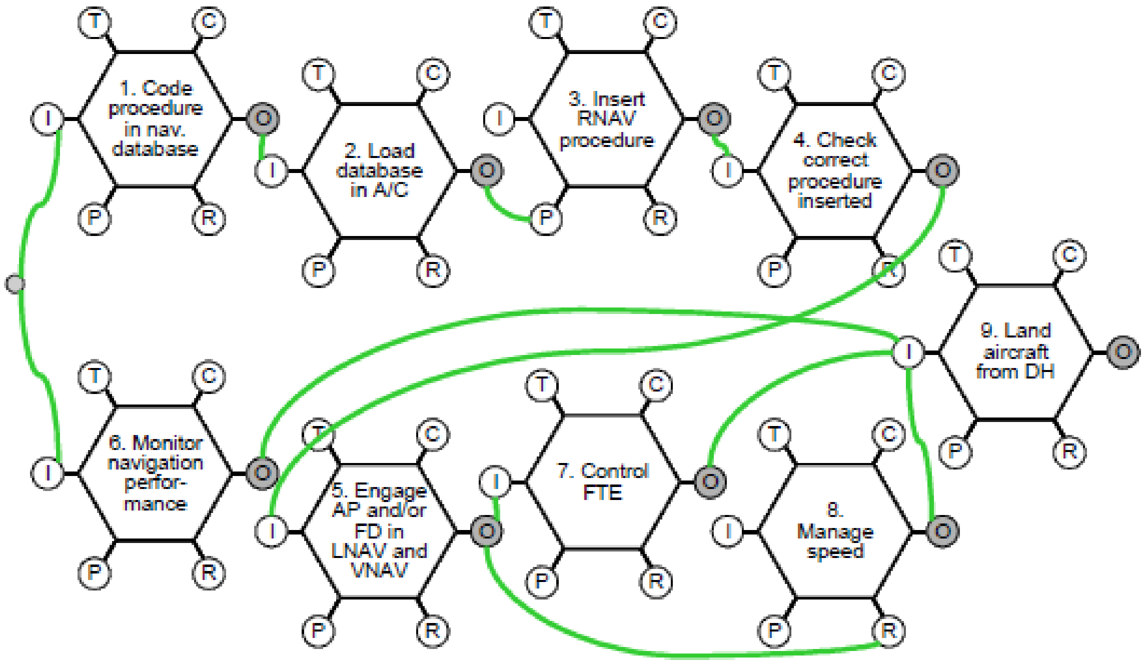

1. Code procedure in nav. database
2. Load database in A/C
3. Insert RNAV procedure
4. Check correct procedure inserted
5. Engage AP and/or FD in LNAV and VNAV
6. Monitor navigation perfor-mance
7. Control FTE
8. Manage speed
9. Land aircraft from DH
T
C
I
O
P
R

Figure 11: FRAM functional representation of aircraft area navigation from [36]

The functional representation models the essential functions of the system and their relationship in terms of six attributes [36]:

- Inputs (I) needed to perform the function
- Outputs (O) produced by the function
- Resources (R) representing what is needed by the function to process the input
- Controls (C) that serve to supervise or restrict the function
- Preconditions (P) specifying system conditions that must be fulfilled before the function is carried out
- Time (T) including actual process duration and permissible time window for the activity

Using the functional representation of a system, the FRAM analysis is conducted in 4 main steps [36]:

1) Identify and characterize essential system functions; the characterization can be based on the six connectors of the hexagonal representation
2) Characterize the (context dependent) potential for variability using a checklist
3) Define functional resonance based on identified dependencies among functions
4) Identify barriers for variability (damping factors) and specify required performance monitoring

Once the functions and their relationships have been identified (step 1), each function is analyzed for potential variability that may be due to contextual influences in the system. Given potential sources for variability in each individual function, the next step looks for ways in which the variability of multiple functions may combine to cause an incorrectly performed or missed function. This includes identifying existing connections in the functional representation that might cause problems as well as new connections that might develop. Finally, barriers and preventative measures are identified to prevent the identified possibilities for functional resonance from causing an accident.

2.2.2.2.1 FRAM Analysis Evaluation

One advantage with this approach is that FRAM analysis defines the type of system representation that the analysis is performed on, as opposed to other methods that rely on a user's internal model of the system. Using a formal system representation may help ensure that engineers and other team members are on the same page with respect to the essential system functions and the behavioral assumptions used in the analysis. The functional representation also emphasizes the importance of interactions and dependencies, which is easily overlooked with other methods.

However, FRAM analysis was developed much more recently than other analysis methods and has only been applied to a few systems. It has been applied to better understand human behavior [51, 125], but it has not yet been applied to study complex software systems in detail. FRAM analysis is also limited to preventing accidents caused by normal variations in performance [50], which captures some causes but omits many types of accidents including those caused when there are no performance variations (e.g. as with deterministic software common in safety-critical systems) or when the variations are abnormal.

2.2.2.3 STPA Hazard Analysis

STPA (System Theoretic Process Analysis) is a hazard analysis method based on the STAMP model of accident causation. STPA has two main steps. STPA Step 1 identifies the potentially unsafe control actions for the control processes in the system being considered. These hazardous control actions are used to create safety requirements and constraints on the behavior of both the system and its components. Additional analysis is then performed to identify additional causal factors and scenarios that can lead to the violation of the safety constraints. As in any hazard analysis, these scenarios are used to control or mitigate the hazards in the system design.

Before beginning an STPA hazard analysis, potential accidents and related system-level hazards are identified along with the corresponding system safety constraints that must be controlled. As an illustrative example, consider a simple automated door control system for a train. The accidents to be considered are: injury to a person by falling out of the train, being hit by a closing door, or being trapped inside a train during an emergency. The system-level hazards for the door control system that are relevant to this definition of accidents include:

H-1: Doors close on a person in the doorway
H-2: Doors open when the train is moving or not in a station
H-3: Passengers/staff are unable to exit during an emergency

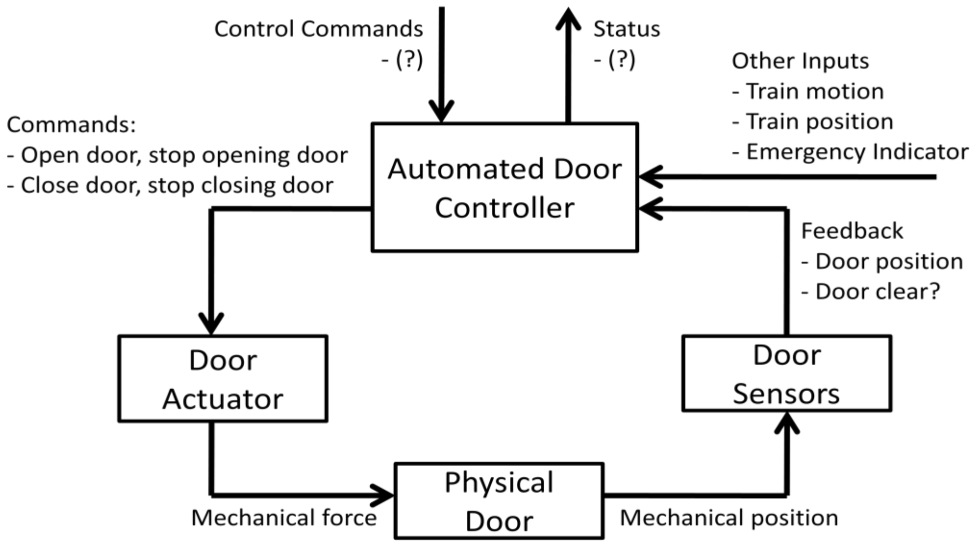


Figure 12: Preliminary control diagram for an automated door controller

STPA is performed on a functional control diagram of the system. Figure 12 shows a simplified control diagram for the train door controller. The first part of STPA identifies hazardous control actions for each component that could produce a system-level hazard by violating the system safety constraints. Once the set of hazardous control actions has been identified, the second part of STPA analyzes the system to determine the potential scenarios that could lead to providing a hazardous control action and scenarios that lead to hazards without hazardous control actions. These scenarios can be used to design controls for the hazards or, if the design already exists, to ensure that these issues are adequately controlled.

STPA Step 1: The first step of STPA identifies control actions for each component that can lead to one or more of the defined system hazards. This step is guided by the four general types of unsafe control actions defined in STAMP:

1. A control action required for safety is not provided or is not followed
2. An unsafe control action is provided that leads to a hazard
3. A potentially safe control action is provided too late, too early, or out of sequence
4. A safe control action is stopped too soon or applied too long

Specific hazardous control actions for each type can be identified and documented using a table as in Table 3. The hazardous control actions can then be translated into system and component safety requirements and constraints.

Table 3: Potentially hazardous control actions for a simple automated door controller

Control Action	**1) Not Providing Causes Hazard**	**2) Providing Causes Hazard**	**3) Wrong Timing or Order Causes Hazard**	**4) Stopped too soon or applied too long**
Provides door open command	Doors not commanded open once train stops at a platform [not hazardous][4] Doors not commanded open for emergency evacuation [see H-3] Doors not commanded open after closing while a person or obstacle is in the doorway [see H-1]	Doors commanded open while train is in motion [see H-2] Doors commanded open while train is not aligned at a platform [see H-2]	Doors commanded open before train has stopped or after it started moving (same as "while train is in motion") [see H-2] Doors commanded open late, after train has stopped [not hazardous] Doors commanded open late after emergency situation [see H-3]	Door open stopped too soon during normal stop [not hazardous] Door open stopped too soon during emergency stop [see H-3]
Provides door close command	Doors not commanded closed or re-closed before moving [see H-2]	Doors commanded closed while person or object is in the doorway [see H-1] Doors commanded closed during an emergency evacuation [see H-3]	Doors commanded closed too early, before passengers finish entering/exiting [see H-1] Doors commanded closed too late, after train starts moving [see H-2]	Door close stopped too soon, not completely closed [see H-2]

Each item in the table should be evaluated to determine whether it is hazardous as defined by the system-level hazards. For instance, in this simple example the doors remaining closed during a routine train stop (non-emergency) is not hazardous because it does not lead to any of the three system-level hazards specified. If this situation is a safety concern, then the hazard list can be updated to include the corresponding hazard. On the other hand, commanding the doors open while the train is in motion is hazardous because it leads to hazard H-2. Each unsafe control action is then

[4] This is not hazardous because it does not lead to any of the system-level hazards (see H-1,H-2,H-3 above). If the hazards and accidents included in the safety analysis were extended to include inconvenience to the passengers, then this item would be considered hazardous.

translated into a component-level safety constraint (e.g. train must not be capable of starting with door open, doors must remain closed while train is in motion, etc.).

STPA Step 2: The second step of STPA examines each control loop in the safety control structure to identify potential causal factors that explain hazardous control actions or otherwise violate the safety constraints.

Figure 13 shows a generic control loop that can be used to guide this step. While STPA Step 1 focuses on the provided control actions (the upper left corner of Figure 13), STPA Step 2 expands the analysis to include causal factors along the rest of the control loop.

Consider a hazardous control action for the automated door controller: the doors are commanded closed while a person is in the doorway. STPA Step 2 would show that one potential cause of that action is an incorrect belief that the doorway is clear (an incorrect process model). The incorrect process model, in turn, may be the result of inadequate feedback provided by a failed sensor, the feedback may be delayed or corrupted. Alternatively, the system may have operated exactly as designed but the designers may have omitted a feedback signal or the feedback requirements may be insufficient.

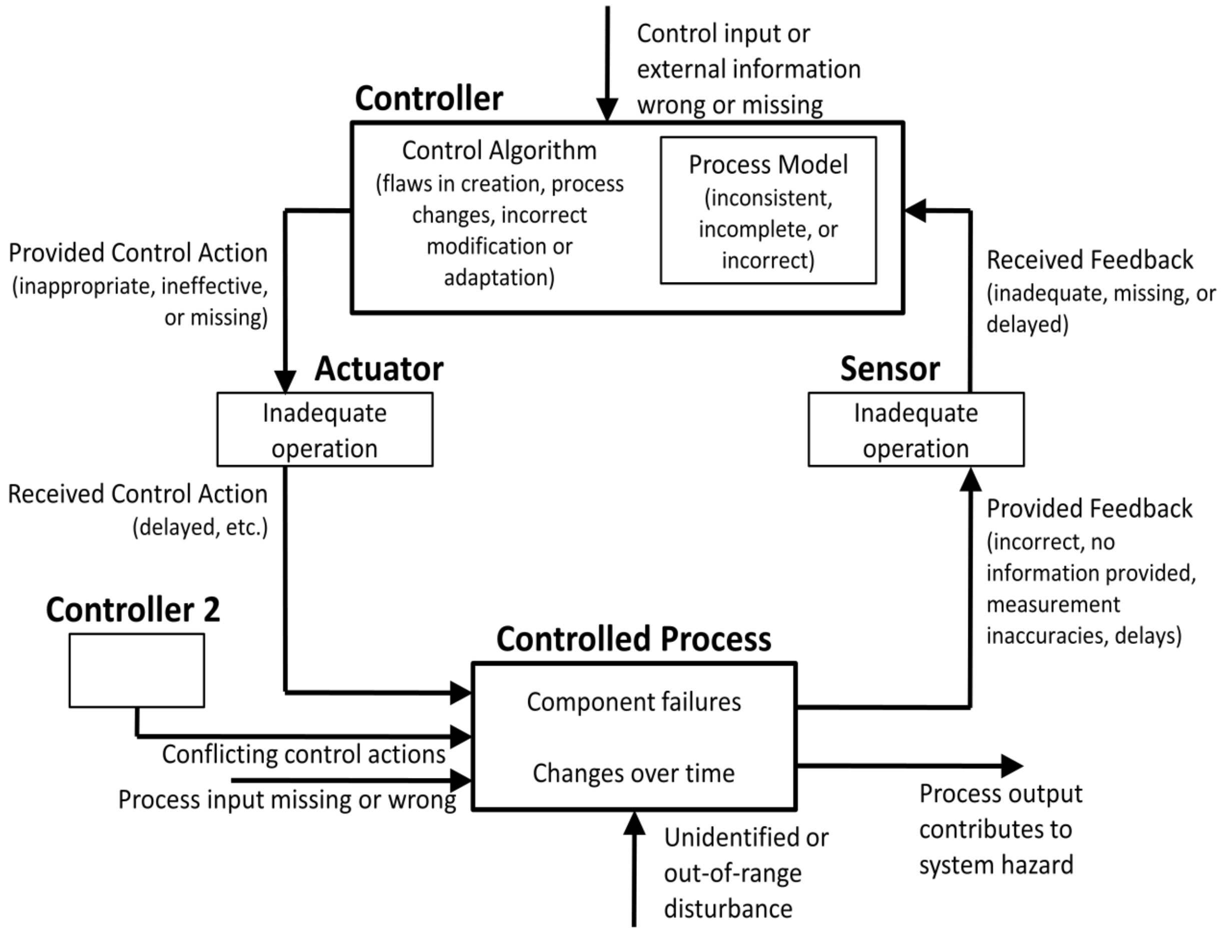

Control input or external information wrong or missing
Controller
Control Algorithm (flaws in creation, process changes, incorrect modification or adaptation)
Process Model (inconsistent, incomplete, or incorrect)
Provided Control Action (inappropriate, ineffective, or missing)
Received Feedback (inadequate, missing, or delayed)
Actuator
Inadequate operation
Sensor
Inadequate operation
Received Control Action (delayed, etc.)
Provided Feedback (incorrect, no information provided, measurement inaccuracies, delays)
Controller 2
Controlled Process
Component failures
Changes over time
Conflicting control actions
Process input missing or wrong
Unidentified or out-of-range disturbance
Process output contributes to system hazard

Figure 13: General control loop with causal factors

Once the second step of STPA has been applied to determine potential causal factors, including causes for each hazardous control action identified in STPA Step 1, the causal factors are eliminated or controlled in the design.

2.2.2.3.1 STPA Evaluation

A significant advantage of STPA is the ability to capture a wide array of causes including organizational aspects, requirements flaws, design errors, complex human behavior, and component failures [126]. The ability to identify requirements flaws is particularly important for software systems because most software-related accidents are caused not by software errors but flawed software requirements [4, 5]. Although many hazard analysis techniques stop once a chain of events or failures has been identified, STPA explains the complex reasons why a sequence of events might occur including underlying processes and control flaws that may exist without any component failure.

Although STPA is a relatively new method, it has been demonstrated on a broad number of systems including aviation systems [32, 127, 128], spacecraft [129-132], organizations[133], missile defense systems [134], and railway systems [135] among others. However, STPA has only been applied in an ad-hoc manner without rigorous procedures, for example, to identify unsafe control actions in a system. In addition, STPA has not yet been formalized mathematically to enable automated methods that could potentially assist in performing the analysis. Moreover, formalization of STPA could be used not only to identify unsafe control actions and other control flaws, but also to generate model-based requirements that will enforce safe behavior.

2.3 Terminology

Accident: An undesired and unplanned (but not necessarily unexpected) event that results in an [unacceptable] level of loss [5]

Component failure accidents: An accident that results from component failures, including the possibility of multiple and cascading failures [10]

Component interaction accident: An accident that arises in the interactions among system components (electromechanical, digital, human, and social) rather than in the failure of individual components [10]

Failure: non-performance or inability of a component to perform its intended function as defined by the component's behavioral requirements [10]

Hazard: A system state or set of conditions that, together with a particular set of worst-case environmental conditions, will lead to an accident (loss) [10]

Safety: The freedom from accidents [5]

Chapter 3. Extending System Theoretic Process Analysis

STPA is a top-down process that begins by identifying the system hazards, the corresponding system safety constraints, and the control structure responsible for enforcing the safety constraints. Given this information, STPA proceeds to identify individual control actions that can violate the system safety constraints and cause hazards. However, thus far the identification of hazardous control actions has been ad-hoc, guided only by the definition of four types of hazardous control actions:

1. A control action required for safety is not provided or is not followed
2. An unsafe control action is provided that leads to a hazard
3. A potentially safe control action is provided too late, too early, or out of sequence
4. A safe control action is stopped too soon or applied too long

This chapter seeks to provide additional guidance to assist in identifying hazardous control actions by defining a general structure for all hazardous control actions. The general structure is then used to develop procedures that can be applied systematically to identify the hazardous control actions in a system. Chapter 4 formalizes this approach mathematically, and develops automated methods that can be used to assist in the analysis.

3.1 General Structure for Hazardous Control Actions

Hazardous control actions are control actions that violate system safety constraints. Table 4 provides examples of control actions and hazardous control actions that might be identified in an STPA analysis. As seen in the examples, a control action by itself does not provide enough information to determine whether it is safe or hazardous—additional information is necessary including the controller providing the control action and the context or environment in which the control action is given. Figure 14 illustrates a generic structure that applies to the hazardous control actions that can be identified.[5]

[5] Note that control actions can be specified in various levels of detail, especially for continuous commands or commands with parameters. However, it is advantageous in a high-level analysis to abstract the command to

Table 4: Examples of hazardous control actions

	Control Action	**Hazardous Control Action**
1	Open train doors	Operator opens train doors while train is moving
2	Capture space vehicle	Space station robotic arm captures space vehicle too early
3	Execute passing maneuver	Pilot executes passing maneuver without clearance
4	Execute passing maneuver	Pilot does not execute passing maneuver upon receiving clearance
5	Execute passing maneuver	Pilot executes passing maneuver too late after receiving clearance

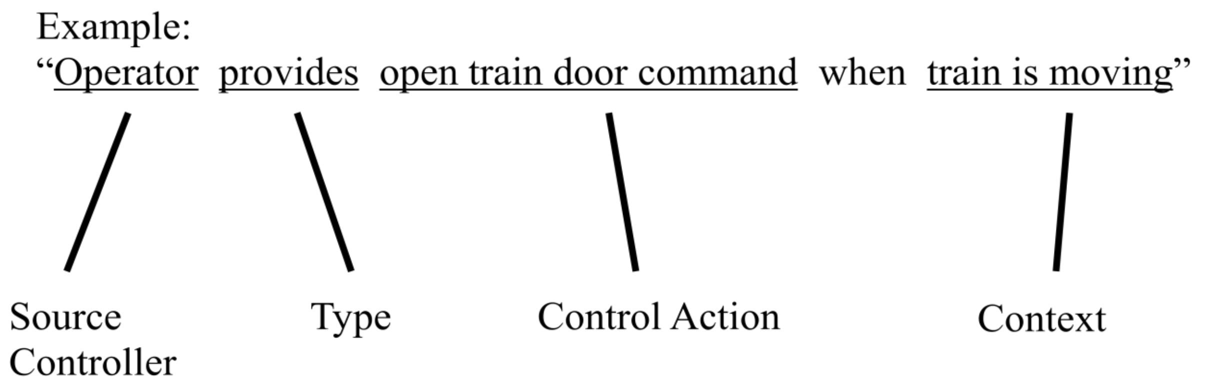

Example:
"Operator provides open train door command when train is moving"
Source Controller
Type
Control Action
Context

Figure 14: Structure of a hazardous control action

The structure in Figure 14 decomposes control actions into four main elements: source controller, type, control action, and context. The *source controller* is a controller that has the capability to provide the specified control action.[6] The *type* of a hazardous control action identifies whether the specified action is provided or not provided—either of which could be hazardous. Finally, the *context* describes the conditions in the system and environment that make action (or inaction) hazardous.

The task of identifying hazardous control actions requires identifying the potentially hazardous contexts of each control action. Although the controllers and control actions are described in the system control structure and are not difficult to identify, hazardous contexts can be more

include only the information necessary to determine if a safety constraint will be violated. More detail can always be added if needed during later iterations and refinements of the analysis.

[6] In more complex control structures with controllers that can issue the same command to multiple entities, a destination field may be included in the hazardous control action statement.

challenging and some contexts may be overlooked using an ad-hoc method. By decomposing the context further it is possible to provide additional guidance and reduce the possibility that important contexts are overlooked. Figure 15 shows how the context for hazardous control actions can be further decomposed into variables and values.

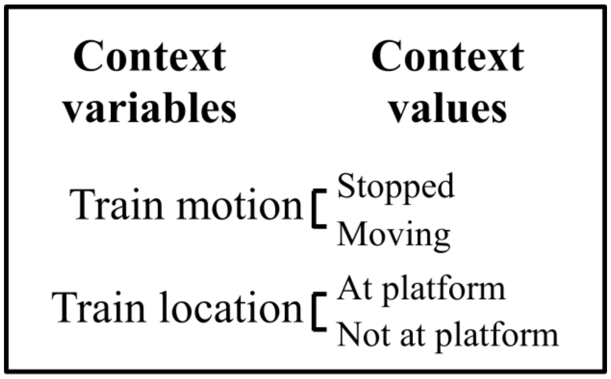


Figure 15: Decomposition of context into variables and values

This simple representation can be much more powerful than identifying hazardous control actions with no guidance. For example, the hazardous control action in Figure 14 only considers "train is moving" but decomposition can help the user to identify additional contexts from other values, like a stopped train, or from other variables, such as train location. The reverse is also true: decomposition can help the user evaluate whether all possible values for a variable have been considered (e.g. stopped and moving) and that it is time to move on to other variables. Another benefit of employing this decomposition is that it reduces ambiguity. For example, consider rows 2 and 5 in Table 4. Although derived ad-hoc and guided by the four types of hazardous control actions, these rows are ambiguous about what constitutes *too early* or *too late* and exactly what makes the control actions hazardous. Using the proposed decomposition, the hazardous control actions can be stated more precisely as "space station captures space vehicle while vehicle thrusters are on" and "pilot executes passing maneuver after clearance has expired". Moreover, ad-hoc approaches may not provide enough information to construct software requirements and procedures. For example, the conclusion "must not capture the vehicle too early" is not very helpful for engineers and operators but the revised statement "must not capture the vehicle while its thrusters are on" is actionable and a number of design solutions can be developed to enforce this behavior.

The structure in Figure 14 can provide further insight by observing that in order to prevent hazardous control actions, at a minimum the source controller needs to be aware of the hazardous contexts. In other words, the controller’s process model must contain at least the variables and values from the hazardous contexts. For STPA applications with an existing design, this provides a way to verify that the design provides controllers with the necessary information—the process model variables must be a superset of the context variables. For safety-driven design processes, this provides a way to use the STPA results to directly define the design parameters—the context variables can be used to define the initial process model and feedback paths in the design, and amended as necessary in later design iterations.

Although this structure provides more guidance than ad-hoc methods, like any hazard analysis technique it is not infallible. This approach still relies on the user’s ability to identify the relevant context variables and values. However, the need to identify hazardous contexts is a task that cannot be effectively avoided with any method; the only choice is whether that task is done indirectly without explicit knowledge or guidance versus performing that task directly with an explicit understanding of the underlying constructs and a framework to systematically reason about missing or complete contexts. Section 3.2 introduces a process that can help users identify the context variables from the system hazards by building a process model hierarchy. Section 3.3 provides a procedure that uses these structures to identify hazardous control actions.

3.2 Process model hierarchy

In systems theory, systems are viewed as hierarchical structures [10]. Although not required, some STPA applications have organized the system hazards into a hierarchy [10, 129]. Hazardous control actions can also be organized into a hierarchy [136]. This section shows how process model variables can also be organized into a hierarchy to provide traceability between the high-level system hazards and individual unsafe control actions in an STPA analysis.

Consider a hypothetical proton therapy machine that treats medical tumors by irradiating a specific area inside a patient. At a high level, one important system hazard occurs if the wrong area of the patient is irradiated, which can lead to a loss of life or health. The corresponding system safety constraint would require that prohibited areas of the patient must not be irradiated. Two system-level conditions or variables can be derived directly from these statements[7]: (1) the area that the machine is about to irradiate and (2) the areas that must not be irradiated. This relationship is shown in Table 5.

Table 5: High-level process model variables for a proton therapy machine

System Hazard H-1	Prohibited area of the patient is irradiated
System Safety Constraint SC-1	Prohibited areas of the patient must not be irradiated
Process model variables	**Irradiated areas**: the areas about to be irradiated **Prohibited areas:** the areas that must not be irradiated

At a system level, controllers must have some knowledge or assumptions about these variables in order to enforce the safety constraint. An STPA analysis must start at this high level, but also must eventually refine these variables for a more detailed analysis of individual controllers within the proton therapy machine, each of which have lower-level responsibilities.

Once the system-level process variables have been identified, refinement can proceed by decomposing each variable using the process functions that govern the relationships between variables. For example, the irradiated area is a function of the patient's position relative to the machine and the location of the beam target. The patient's position, in turn, is a function of the table position (for example, controlled by a set of servos and software programs) and the patient position on the table (for example, controlled by an immobilization device like a custom body cast or bite mold). These safety-critical process variables and their relationships can be organized into an STPA process model hierarchy as follows:

[7] This relationship exists in general and is actually a natural result of the definition of a hazard used in STPA and STAMP; a hazard is a *system state or set of conditions* that, together with a particular set of worst-case environmental conditions, will lead to an accident (loss).

Process Model Hierarchy

- Irradiated Area
    - Patient Position
        - Table Position
        - Patient Position on Table
    - Beam Target
        - Beam energy
        - Beam shape
        - Beam trajectory
    - Beam dose
        - Beam intensity
        - Time at target
- Protected Areas (defined by radiation oncologist)

The identification of potentially hazardous contexts and hazardous control actions can then proceed at lower levels of refinement guided by the process model hierarchy. For example, low-level hazardous control actions like “Table controller does not stop linear X servo when table position reaches gantry coupling” could be identified. Figure 16 shows an example of the relationships and traceability that can be established between system hazards, the process model hierarchy, and hazardous control actions.

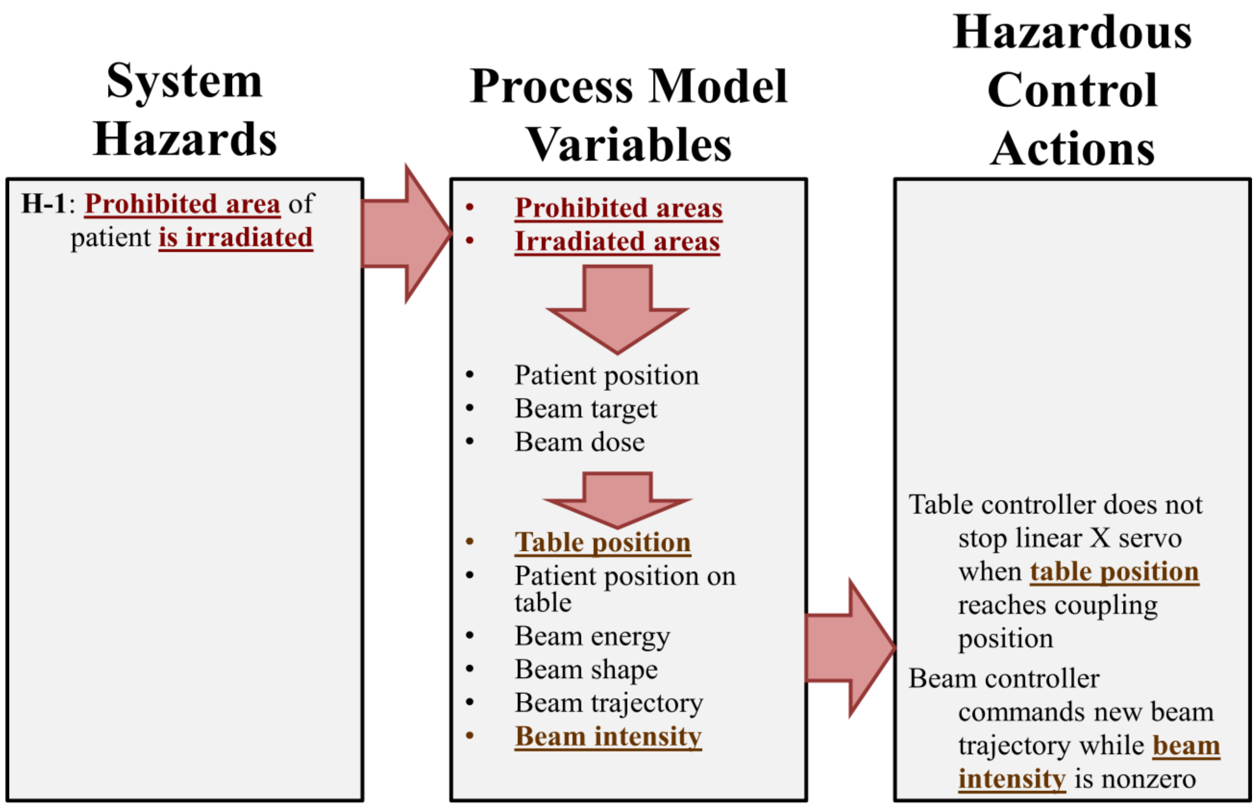

System Hazards
H-1: Prohibited area of patient is irradiated
Process Model Variables
• Prohibited areas
• Irradiated areas
• Patient position
• Beam target
• Beam dose
• Table position
• Patient position on table
• Beam energy
• Beam shape
• Beam trajectory
• Beam intensity
Hazardous Control Actions
Table controller does not stop linear X servo when table position reaches coupling position
Beam controller commands new beam trajectory while beam intensity is nonzero

Figure 16: Example of traceability using a process model hierarchy of a medical device

This proton therapy example presented here has been simplified for demonstration purposes, however a more complete STPA analysis of an actual proton therapy machine using the methods developed in this chapter can be found in [136].

As another example of a process model hierarchy, consider a new aircraft passing procedure [137]. One of the most important system hazards occurs if aircraft come to close to each other and violate minimum separation rules. To enforce minimum separation, at a high level there are three process variables that must be known or controlled: (1) the current separation, or relative position, of the aircraft, (2) the required separation, and (3) the future or anticipated separation. These high-level process variables can be decomposed into a hierarchy of process model variables to guide the STPA analysis (and the system design process):

Process Model Hierarchy

- Current Separation
    - X,Y, Z position for each aircraft
- Required Separation
    - Lateral Requirement
    - Vertical Requirement
- Future Separation
    - X, Y, Z position for each aircraft
    - Current trajectory for each aircraft
    - Flight plan for each aircraft

By defining the process model hierarchy, potentially hazardous contexts can be derived from the system hazard definitions and used to identify lower-level hazardous control actions in STPA. Figure 17 shows an example of low-level contexts in hazardous control actions that can be derived from (and traced to) the high-level system hazards. By establishing this traceability, the process model hierarchy can be used to update relevant parts of the analysis when design changes are made or when the system is used in a new environment and environmental assumptions change. Rather than repeating the entire analysis, the hierarchy can reveal which low-level aspects are impacted by new or different hazards. The traceability in a process model hierarchy can also help to review the completeness and consistency of the analysis. A more complete STPA analysis of this system, which is based on actual NextGen In-Trail Procedures, can be found in [127].

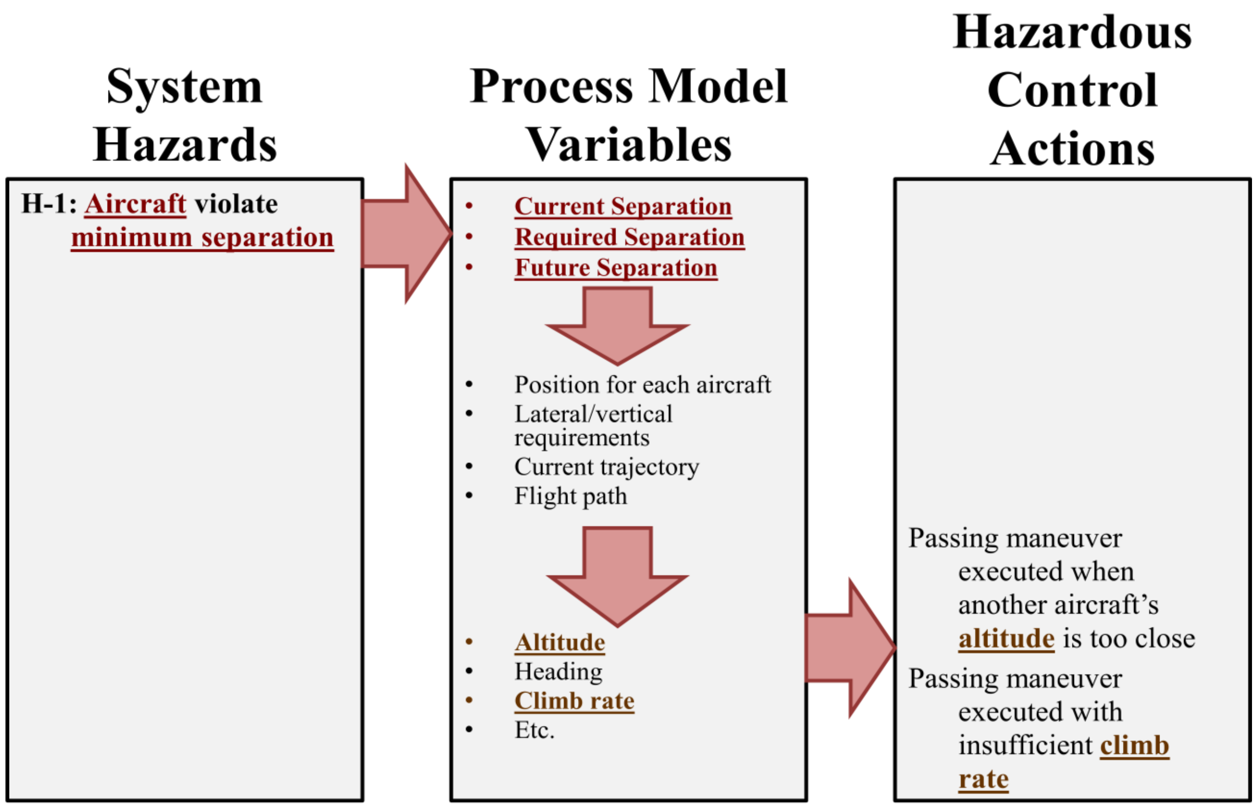

System Hazards
Process Model Variables
Hazardous Control Actions
H-1: Aircraft violate minimum separation
• Current Separation
• Required Separation
• Future Separation
• Position for each aircraft
• Lateral/vertical requirements
• Current trajectory
• Flight path
• Altitude
• Heading
• Climb rate
• Etc.
Passing maneuver executed when another aircraft's altitude is too close
Passing maneuver executed with insufficient climb rate

Figure 17: Aviation example of traceability using a process model hierarchy

3.3 A Systematic Procedure to Identify Hazardous Control Actions

This section introduces a new procedure for rigorously and systematically identifying the hazardous control actions during the first step of STPA [138]. The procedure is based on the hazardous control action structure described in section 3.2, and involves identifying potential control actions and hazardous states and then analyzing which combinations yield a hazardous control action.

Two parts of the procedure are described in the following sections, and each part can be performed independently of the other. The first part deals with control actions that are provided under conditions that make the action hazardous. The second part deals with control actions that are *not* provided under conditions that make *inaction* hazardous.

A simplified train example is used to introduce the procedure, followed by a more complex demonstration with a nuclear power plant application. The simplified train example analyzes the train door control loop, including the door controller. The process is applicable to early development phases before any detailed design information exists, and the identified hazardous control actions apply whether the door controller is ultimately implemented as a human operator or as an automated software program. The hazards for the example train door controller are as follows:

H-1: Doors close on a person in the doorway
H-2: Doors open when the train is moving or not in a station
H-3: Passengers/staff are unable to exit during an emergency

The example control structure used to introduce the procedure is shown in Figure 18.

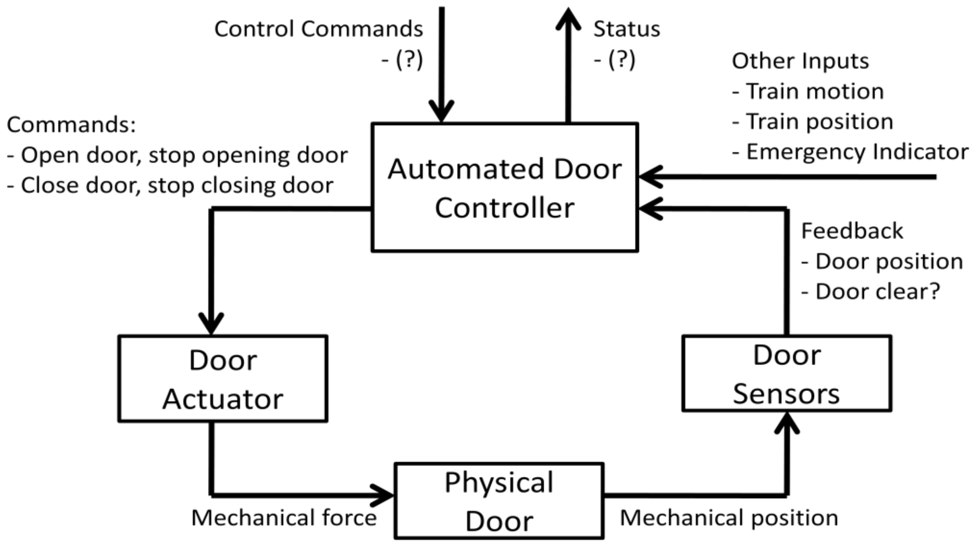


Figure 18: Partial control structure for simplified train door controller

3.3.1 Part 1: Control actions provided in a state where the action is hazardous

The first part of the procedure is to select the controller and the associated control actions from the control structure. In the train example above, the automated door controller can provide four control actions: *open doors*, *stop opening doors*, *close doors*, *or* *stop closing doors*.[8] Although the *open door* command is analyzed in the following examples, the same procedure can be applied to the other control actions.

Next, the controller's process model is defined to determine the environmental and system states that affect the safety of the control actions. As discussed earlier, the required variables in the process model can be derived from the system hazards defined at the start of an STPA analysis. For example, hazard H-1 identifies the state of the doorway (whether it is clear or not) as an important environmental variable in deciding whether to close the doors. Figure 19 shows the required process model for the door controller.

[8] Note that when the controller has the ability to command the stopping of some process, that command is also a control action and must be analyzed. In this way, continuous hazardous control actions related to "stopped too soon" and "applied too long" are explicitly covered by this procedure. In fact, "applied too long" was not included in early exploratory applications of STPA [128] but was added to STPA in 2011 [138] when the procedure described here identified new hazardous control actions that had not been found previously.

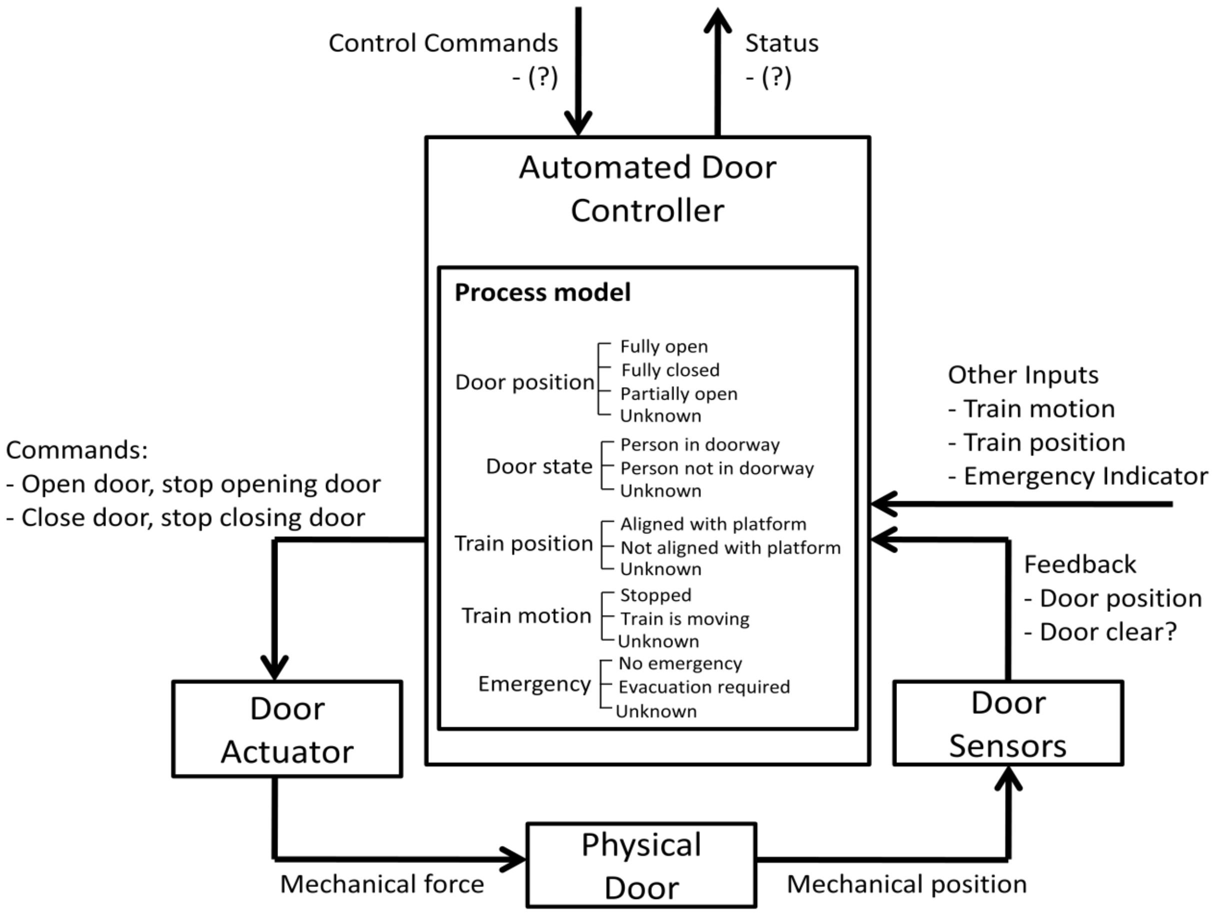


Figure 19: Augmented control structure with the door controller's process model

Once the process model variables have been identified, potentially hazardous control actions can be identified by examining each combination of relevant process model values and determining whether issuing the control action in that state will be hazardous. For example, one possible context for the <u>*open door*</u> command consists of the values: the train is stopped, there is no emergency, and the train is not aligned with a platform. Providing the <u>*open door*</u> command in this context is a hazardous control action.

Each row in Table 6 specifies a different context for the <u>*open door*</u> command. Context here is defined as a combination of values of the process model variables. Each context can be evaluated to determine whether the control action is hazardous in that context, and the result is recorded in the three columns on the right. The two right-most columns incorporate timing information as well.[9] For example, providing an *open door* command in the context of an emergency while the train is stopped is not hazardous; in fact, that's exactly what should happen for evacuation purposes. However, providing the *open door* command *too late* in that context is certainly hazardous.

Table 6: Context table for the *open door* control action

Control Action	**Train Motion**	**Emergency**[10]	**Train Position**	**Hazardous control action?**		
				If provided any time in this context	**If provided too early in this context**	**If provided too late in this context**
Door open command provided	Train is moving	No emergency	(doesn't matter)	Yes	Yes	Yes
Door open command provided	Train is moving	Emergency exists	(doesn't matter)	Yes[11]	Yes	Yes
Door open command provided	Train is stopped	Emergency exists	(doesn't matter)	No	No	Yes
Door open command provided	Train is stopped	No emergency	Not aligned with platform	Yes	Yes	Yes
Door open command provided	Train is stopped	No emergency	Aligned with platform	No	No	No

Note that during this process, some combinations of conditions may expose conflicts in the design that need to be considered. For example, is it hazardous to provide the *open door* command during a fire (an emergency) while the train is in motion? In other words, is it safer to keep the doors closed and trap the passengers inside or is it better to open the doors and risk physical injury because the train is moving? These questions can and should prompt exploration outside the automated door controller. For example, the issue might be addressed in the design by providing a way for

[9] Although some techniques treat "too late" and "too early" actions as subsets or combinations of broader categories like *not provided* (i.e. omission) and *provided incorrectly* (i.e. commission), we have found that keeping these classes explicit often helps analysts consider more types of hazardous control and identify more accident causes than when "too late" and "too early" are only implicitly included.

[10] Note that emergency is defined as in Figure 19, that is, a condition that requires evacuation of the train

[11] This row is an example of a conflict; see chapter 4 for more information.

passengers to exit to nearby train cars when there is an emergency and the train is moving. In addition, the braking system controller can be designed to apply the brakes in that context (emergency and train is moving) to minimize the duration of that hazardous situation.

3.3.2 Part 2: Control actions not provided in a state that makes inaction hazardous

It is also necessary to consider potential contexts in which the lack of a control action is hazardous. The same basic process is used: identify the corresponding process model variables and the potential values, create contexts for the action using combinations of values, and then consider whether an absence of the specified control action would be hazardous in the given context. Table 7 shows the identification of hazardous control actions for the door open command not being provided.

Table 7: Context table for the lack of an *open door* control action

Control Action	**Train Motion**	**Emergency**	**Train Position**	**Door State**	**Hazardous if not provided in this context?**
Door open command not provided	Train is stopped	No emergency	Aligned with platform	Person not in doorway	No[12]
Door open command not provided	Train is stopped	No emergency	Not aligned with platform	Person not in doorway	No
Door open command not provided	Train is stopped	No emergency	Aligned with platform	Person in doorway	Yes
Door open command not provided	Train is stopped	No emergency	Not aligned with platform	Person in doorway	No[13]
Door open command not provided	Train is stopped	Emergency exists	(doesn't matter)	(doesn't matter)	Yes
Door open command not provided	Train is moving	(doesn't matter)	(doesn't matter)	(doesn't matter)	No

Again, some combinations of conditions are uncovered that expose potential conflicts and need to be considered in the design. For example, is it hazardous to provide the *open door* command when the train is stopped away from a platform and a person is in the doorway? Although every effort should be made to prevent this context from happening, it may be conceivable; for example, perhaps the train can leave the platform after a door closes on a person or their belongings. If a person is trapped away from a platform, is it safer to open the door or keep it closed? These questions can lead to exploration outside the automated door controller; for example, this issue might be addressed by ensuring a crew member will be alerted to assist the passenger. In terms of the door controller, for the purpose of this simple demonstration it is assumed that it is best to keep the door closed to prevent a potentially trapped passenger from falling out of the train before assistance arrives.

[12] This row is not hazardous because it does not lead to any of the system-level hazards (see H-1,H-2,H-3 in the previous section). If the hazards and accidents included in the safety analysis were extended to include inconvenience to the passengers, then this row would describe a hazardous control action.

[13] For the purpose of this analysis it is assumed that in this case it is best to keep the door closed and alert a crew member to assist the potentially trapped passenger.

The procedures described here represent a “brute-force” approach and, although they provide a systematic approach with more guidance than previous ad-hoc methods, they can be time-consuming when applied to low-level contexts with many variables and values. To address this issue, chapter 4 proposes automated algorithms that can be used to assist in the analysis and chapter 5 proposes both manual and automated techniques that can be employed to reduce the amount of effort required to analyze extremely complex systems.

The resulting hazardous control actions can be summarized in a table based on the four types of hazardous control actions defined in STAMP, as shown in Table 8.

Table 8: Hazardous control actions for the Part 1 and Part 2 context tables

Control Action	**Hazardous Control Actions**			
	Not Providing Causes Hazard	**Providing Causes Hazard**	**Wrong Timing or Order Causes Hazard**	**Stopped Too Soon or Applied Too Long**
Open train doors	Door open command not provided when train is stopped at platform and person in doorway Door open command not provided when train is stopped and emergency exists	Door open command provided when train is moving and there is no emergency Door open command provided when train is moving and there is an emergency[14] Door open command provided when train is stopped unaligned with platform and there is no emergency	Door open command is provided more than X seconds after train stops during an emergency	N/A

[14] To resolve this conflict, a design decision could be made to allow passengers to evacuate to other train cars in this situation while ensuring that the brakes are applied so that evacuation from the train will soon be possible.

3.3.3 Application to a nuclear power plant system

The proposed procedure has been applied successfully to complex systems including a medical proton therapy machine [136] and nuclear power plant systems [139, 140]. This section demonstrates how the procedure can be used to identify hazardous control actions related to a critical and partially-automated part of a nuclear power plant. The appendix contains a longer STPA analysis of the nuclear power plant system and demonstrates how the concepts developed throughout this dissertation can be integrated into a comprehensive STPA-based hazard analysis.

3.3.3.1 System description

The system analyzed is a generalized version of an EPR (Evolutionary Power Reactor), which is a type of PWR (Pressurized Water Reactor). The EPR reactor is fully digital, that is, all control systems, including the Reactor Protection System, are digital. The analysis focuses on one Steam Generator (SG), one Main Steam Line (MSL), and the systems involved in closing the Main Steam Isolation Valve (MSIV), although the same process could be applied to the rest of the system.

A generic diagram of a PWR is shown in Figure 20. During normal operation, the coolant in the primary cooling system (left of the diagram) transfers heat from the reactor to the Steam Generator (SG). The SG contains water that cools the primary coolant and evaporates into steam. The SG prevents primary coolant, which is radioactive, from mixing with the water, which is not radioactive. The steam produced in the SG travels to a turbine connected to a generator to produce electricity. The steam is cooled in the condenser and pumped back into the SG to begin the cycle again. The loop formed by the SG, turbine, and condenser is known as the secondary cooling system.

The MSIV is a valve located on the main steam line from the SG. During normal operation, the MSIV is kept open to permit cooling of the primary cooling system via the secondary system. In case of an abnormal situation, the MSIV can be closed to isolate the SG from the rest of the secondary system. MSIV closure is necessary if there is a break in the main feedwater pipe to the SG that allows water to leak out, an internal SG Tube Rupture (SGTR) that allows primary coolant to mix with secondary water, or a break in the main steam line exiting the SG.

Because MSIV closure prevents the secondary system from adequately cooling the primary system, a number of backup systems are provided to cool the primary coolant in case of MSIV closure. These backup systems include redundant SGs, turbine bypass valves, main steam relief isolation valves (MSRIV) and main steam relief control valves (MSRCV), safety relief valves (SRV), the Chemical Volume Control System (CVCS), and the Emergency Core Cooling System (ECCS). These systems are included in the analysis only to the extent that they impact the decision to close the MSIV.

-

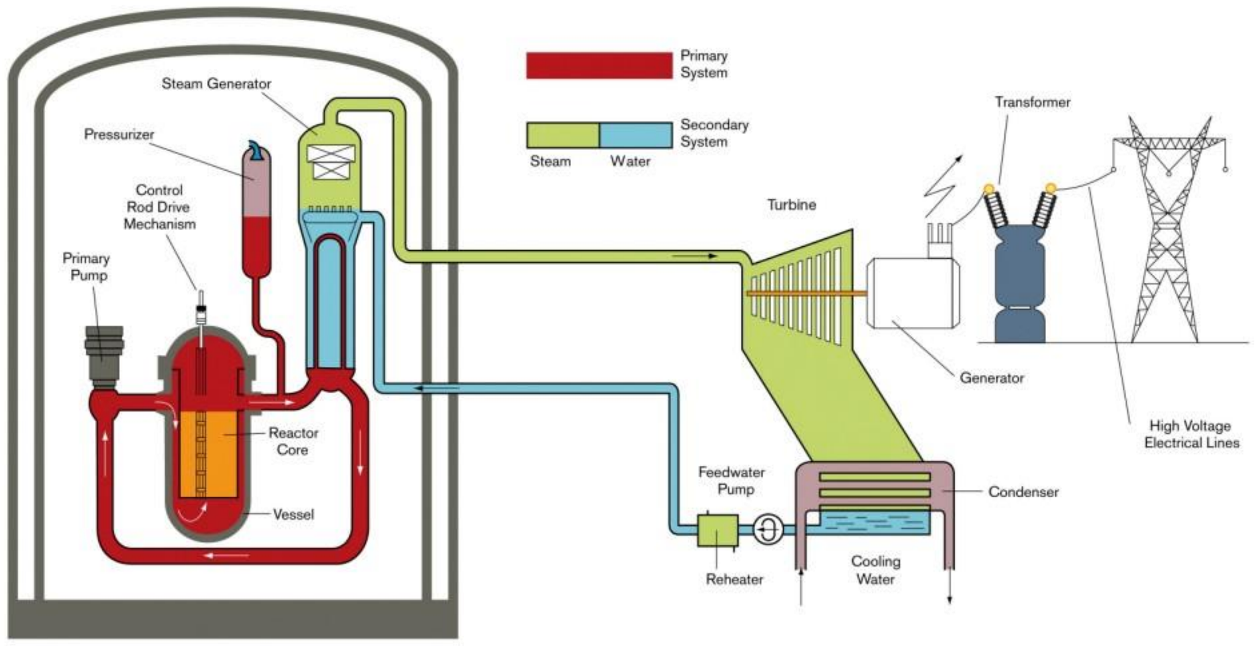


Figure 20: Pressurized Water Reactor (Diagram from [141])

The hazards for the nuclear power plant are described in Table 9. See the appendix for a more detailed description of the accidents and hazards for this system.

Table 9: System hazards for the nuclear power plant

Hazards
H-1: Release of radioactive materials
H-2: Reactor temperature too high
H-3: Equipment operated beyond limits
H-4: Reactor shut down

The high-level safety control structure is shown in Figure 21. The dotted (green) arrow represents the communication between the MSIV controllers and other controllers. For example, the Protection System (PS) contacts the Safety Control System (SCS) in order to initiate the Engineering Safety Features (ESF) controls following ESF actuation. The Reactor Controls (RC) controller also communicates with Non-Safety System Controller (NSSC) in order to provide command signals for actuators used in RC functions other than control rods, such as the BMC (Boron and Makeup Control) components for Boron control.

There are four controllers that can provide a control action to close the MSIV: the Operator, the Non-Safety System Controller (NSSC), the Protection System (PS), and the Diverse Automation System (DAS). These four controllers send control actions to the MSIV Priority Module (PM), which uses a pre-programmed priority setting to determine which control actions to forward to the MSIV actuator. In this sense, the PM can also send control actions.

If the operator detects a need to close the MSIV, he or she may issue a *Close MSIV* command to the PM. The PM determines which controller is in charge according to a priority scheme, and forwards commands directly to the MSIV actuator. In this case, the PM would normally forward the command from the operator to the MSIV actuator. The operator may also send a *Close MSIV* command to the NSSC, which provides manual control for the MSIV. In this situation, the NSSC would normally forward the command from the operator to the PM, which would then forward the command to the MSIV actuator.

The PS is an automated system that can automatically detect some situations in which a *Close MSIV* command is necessary. In these situations the PS can provide the *Close MSIV* command to the PM, which can forward the command to the MSIV actuator. The DAS (Diverse Actuation System) is also an automated system, used as a backup in case there is a problem with the PS. The DAS can issue a *Close MSIV* command to the PM, which would normally forward the command to the MSIV actuator.

A sensor provides feedback about the MSIV status directly to the PM. This sensor does not sense process variables such as pressure, temperature, or steam flux. Instead, it senses torque applied to the valve itself to detect if the valve has closed. The PM receives this feedback and can provide confirmation back to the controller that originally requested the MSIV closure. Other sensors report process variables to the controllers including various pressures, SG water level, and the operation of other backup systems. This information is used by the controllers to determine whether the MSIV should be closed.

See the appendix for more information about the controllers, control algorithms, and responsibilities assigned in this system.

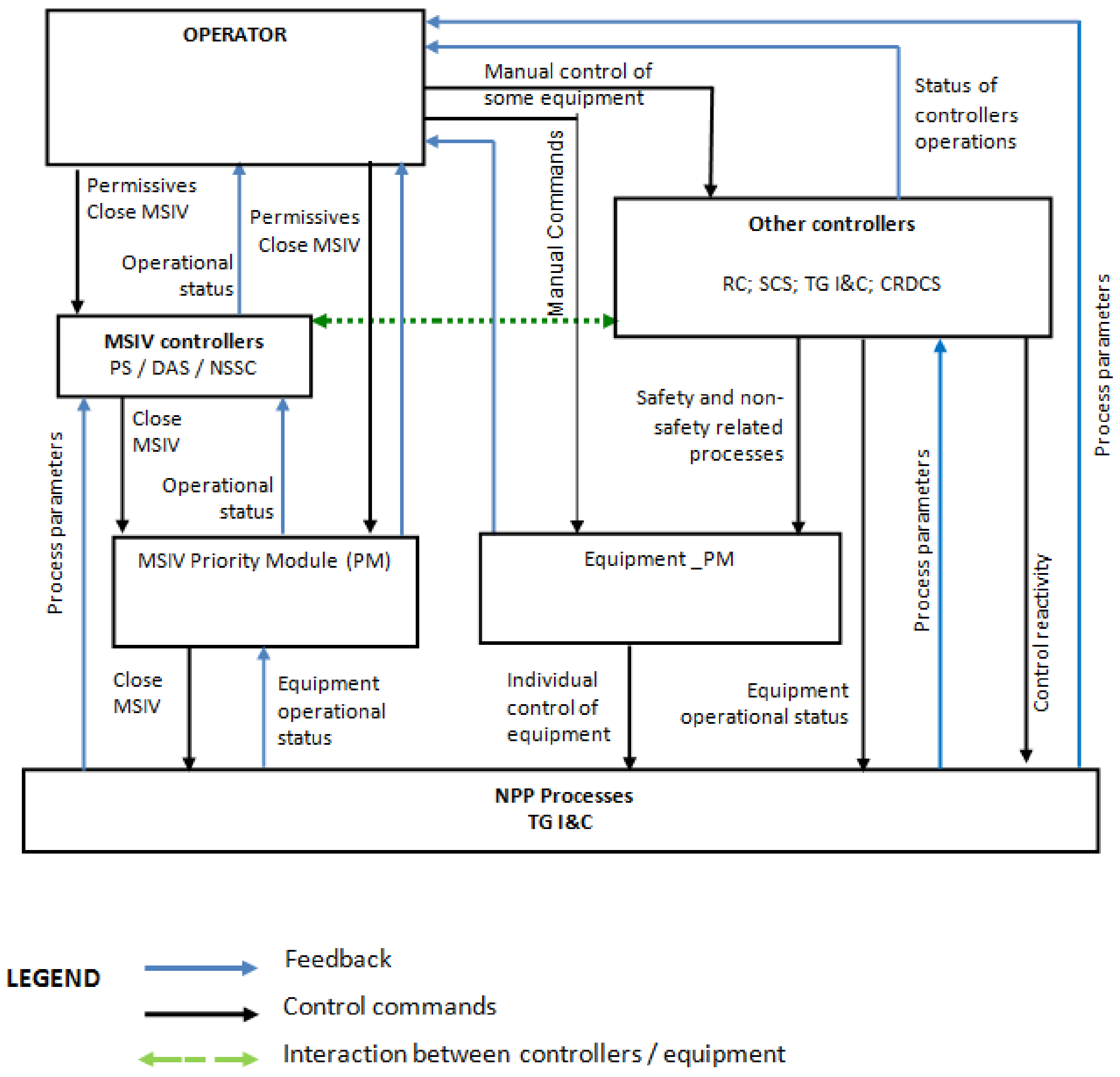


Figure 21: High-level control structure for the nuclear power plant

3.3.3.2 Identifying Hazardous Control Actions

The high-level process model variables associated with MSIV closure can be identified by considering the purpose of the MSIV. The MSIV remains open during normal plant operation and is only needed to control a few specific abnormal conditions. The relevant high-level conditions can be derived from the system hazards and system description as follows: [15]

- Steam generator tube rupture (can cause an uncontrolled SG level increase and can release contaminated fluid into the secondary system)
- Steam system piping leak (can depressurize the SG and cause an overcooling transient and energy release into containment)
- Feedwater system piping leak (can depressurize the SG and cause an overcooling transient and energy release into containment)

While these conditions could be caused by physical failures, the latter two could also be caused by design flaws or unsafe commands elsewhere in the system. For example, a leak in the main steam line could be caused by a physical failure (e.g. rupture in the line) or it could be caused by main steam relief valves that are opened inadvertently or at the wrong time. Both situations could require MSIV closure to prevent depressurization and an overcooling transient while the issue is investigated and resolved.

In addition to helping to mitigate the conditions above, the MSIV also controls the heat exchange that takes place within the SG. Before the MSIV is closed, other support systems**[16]** may need to be engaged to provide the additional cooling needed. Therefore, information about additional cooling provided by other support systems (i.e. inadequate, adequate[17]) may be needed for the decision to close the MSIV and should be included in the process model.

[15] See also [142] chapter 7 pages 7.3-22 and 7.3-11

[16] *Other support systems* refers to other components designed to cool the primary system. These include the CVCS, SI, CCS, etc.

[17] *Adequate* means the system operation is sufficient to provide the cooling normally provided by the SG.

When considering whether a potential control action is hazardous or not, it is important to avoid assuming that other defense barriers are intact or that they are appropriate, sufficient, and error-free. For example, even if there is an emergency feedwater system to provide the necessary cooling in the event of a relief valve inadvertently commanded open, it is still hazardous to inadvertently command the relief valve open. These hazardous actions must be included in a safety analysis and prevented regardless of other protective systems intended to mitigate unsafe behavior.

Table 10 summarizes the hazardous control actions that were identified for the command *Close MSIV* using the systematic procedure.

Table 10: Hazardous Control Actions for Close MSIV

Control Action	**Hazardous Control Actions**			
	Not Providing Causes Hazard	**Providing Causes Hazard**	**Wrong Timing or Order Causes Hazard**	**Stopped Too Soon or Applied Too Long**
Close MSIV	Close MSIV not provided when there is a rupture in the SG tube, leak in main feedwater, or leak in main steam line [H-2, H-1, H-3]	Close MSIV provided when there is no rupture or leak [H-4] Close MSIV provided when there is a rupture or leak while other support systems are inadequate [H-1, H-2, H-3]	Close MSIV provided too early (while SG pressure is high): SG pressure may rise, trigger relief valve, abrupt steam expansion [H-2, H-3] Close MSIV provided too late after SGTR: contaminated coolant released into secondary loop, loss of primary coolant through secondary system [H-1, H-2, H-3] Close MSIV provided too late after main feedwater or main steam line leak [H-1, H-2, H-3, H-4]	N/A

The hazardous control actions in Table 10 were identified using the following process. First, a controller and control action were selected. The operator and the control action *Close MSIV* were analyzed first, although the results also apply to other controllers in the system. A context table was then constructed for the control action using the corresponding process model variables that were defined previously. Table 11 shows the context table for *Close MSIV provided*.

Table 11: Context table for *Operator provides Close MSIV* control action

	1	2	3	4	5	6	7	8
	Control Action	**Steam Generator Tube**	**Condition of Main Feedwater Pipe**	**Condition of Main Steamline**	**Operation of other support systems**	**Control Action Hazardous?**	**Control Action Hazardous if Too Late?**	**Control Action Hazardous if Too Early?**
1	*Close MSIV*	Not Ruptured	No Leak	No Leak	Adequate	H-4	H-4	H-4
2		Ruptured	No Leak	No Leak	Adequate	No	H-1, H-2, H-3, H-4	H-3, H-4
3		Not Ruptured	Leak	No Leak	Adequate	No	H-2, H-3, H-4	No
4		Not Ruptured	No Leak	Leak	Adequate	No	H-2, H-3, H-4	No
5		Ruptured	Leak	No Leak	Adequate	No	H-1, H-2, H-3, H-4	H-3, H-4
6		Not Ruptured	Leak	Leak	Adequate	No	H-2, H-3, H-4	No
7		Ruptured	No Leak	Leak	Adequate	No	H-1, H-2, H-3, H-4	H-3, H-4
8		Ruptured	Leak	Leak	Adequate	No	H-1, H-2, H-3, H-4	H-3, H-4
9		Not Ruptured	No Leak	No Leak	Inadequate	H-2, H-4	H-2, H-4	H-2, H-4
10		Ruptured	No Leak	No Leak	Inadequate	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4
11		Not Ruptured	Leak	No Leak	Inadequate	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4
12		Not Ruptured	No Leak	Leak	Inadequate	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4
13		Ruptured	Leak	No Leak	Inadequate	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4
14		Not Ruptured	Leak	Leak	Inadequate	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4
15		Ruptured	No Leak	Leak	Inadequate	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4
16		Ruptured	Leak	Leak	Inadequate	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4

Column 1 in Table 11 is the control action being analyzed while columns 2 to 5 correspond to the process model variables identified previously. Column 6 specifies in which contexts it is hazardous to provide the *Close MSIV* control action. For example, row 1 describes a situation in which it is hazardous to close the MSIV: if there is no SG tube rupture, no main feedwater pipe leak, and no main steam line leak, then there is no need to close the MSIV. Closing the MSIV will cause H-4 (reactor shut down). If the operation of other support systems cannot make up for the additional heat exchange required, closing the MSIV will also lead to a loss of necessary cooling (H-2 in row 9 column 6).

If other support systems, including other CVCS, SI, ECCS, etc., are producing the additional cooling required during a rupture/leak, then closing the MSIV is not hazardous (rows 2-8, column 6) and a reactor shutdown is initiated regardless of any MSIV actions. If for some reason the other systems are not capable of producing the additional cooling needed, then closing the MSIV may cause other hazards (rows 10-16, column 6) including excessive temperature increase (H-2), release of radioactive materials (H-1), an immediate reactor shutdown or SCRAM (H-4) if not already triggered, and additional equipment damage (H-3). Depending on the type of rupture/leak, it may actually be better to keep the MSIV open to control the temperature of the reactor (H-2) even though that would permit some radioactive steam to be introduced into the secondary system (H-1).

The last two columns on the right in Table 11 take into account timing information. If there is a rupture/leak and other support systems are adequate, then it is not hazardous to close the MSIV (rows 2-8). The MSIV should be closed. However, if the MSIV is closed *too late* in this context then it is hazardous. If the steam generator tube is ruptured, too much radioactive coolant may have already been released into the secondary system and the environment (H-1). If the steam line has a leak, excessive steam may have been released causing overcooling and overcompensation (H-2). If the steam line or feedwater pipe have a leak, the SG may run dry and cause equipment damage (H-3). Closing the MSIV *too early* may also be hazardous in some situations. For example, if the steam generator tube is ruptured then the SG pressure should be decreased before the MSIV is closed. Otherwise, if the MSIV is closed too early after a SG tube rupture, then the SG pressure and temperature will increase and may cause equipment damage to the SG, SG piping, or other systems (H-3).

The contexts used to define hazardous control actions may not be the same as contexts that are inherently unsafe. The tables in this section are used to analyze controller behavior and control actions in a number of contexts, not to analyze contexts that are unsafe by themselves. For example, row 1 column 6 of Table 11 is marked as hazardous because the control action *Close MSIV* will cause a hazard if provided in that context, even though the context by itself (no ruptures/leak) does not describe anything hazardous. Conversely, the context in row 2 describes a steam generator tube rupture but column 6 is not marked as hazardous because closing the MSIV is not a hazardous behavior in that context. In fact, closing the MSIV is exactly what should happen in that situation to prevent an accident.

Although providing a control action can be hazardous, *not* providing a control action can be equally hazardous. Table 12 shows the context table for *not* providing the *Close MSIV* control action. As before, a reactor shutdown should be initiated for any rupture regardless of the MSIV control action. However because these tables are used to identify hazardous control actions, only hazards that are affected by an absent *Close MSIV* control action are listed at this stage of the analysis.

If there is no rupture/leak, keeping the MSIV open is not hazardous (rows 1 and 9). However, if there is a rupture, different hazards may be experienced depending on what part of the system is affected. If the SG tube is ruptured and the MSIV is not closed, radioactive material will be released into the secondary system (H-1) and the SG water level may increase uncontrollably. A sustained release of primary coolant will decrease the effectiveness of the primary cooling system (H-2), and the release of radioactive material into the secondary system may cause equipment damage (H-3). If the main steam line has a leak and the MSIV is not closed, excessive steam may be released causing an overcooling transient and overcompensation by other systems to increase reactivity (H-2). Excessive steam release may also lower the SG water level, causing potential equipment damage if the SG runs dry (H-3). If the main feedwater pipe has a leak and the MSIV is not closed, the SG may be depressurized causing an overcooling transient and water level may drop, leading to H-2 and H-3 as above.

Table 12: Context table for *Close MSIV control action is not provided*

	1	2	3	4	5	6
	Control Action	**Steam Generator Tube**	**Condition of Main Feedwater Pipe**	**Condition of Main Steamline**	**Operation of other support systems**[18]	**Not Providing Control Action is Hazardous?**
1	*Close MSIV*	Not Ruptured	No Leak	No Leak	Adequate	No
2		Ruptured	No Leak	No Leak	Adequate	H-1, H-2, H-3, H-4
3		Not Ruptured	Leak	No Leak	Adequate	H-2, H-3
4		Not Ruptured	No Leak	Leak	Adequate	H-2, H-3
5		Ruptured	Leak	No Leak	Adequate	H-1, H-2, H-3, H-4
6		Not Ruptured	Leak	Leak	Adequate	H-2, H-3
7		Ruptured	No Leak	Leak	Adequate	H-1, H-2, H-3, H-4
8		Ruptured	Leak	Leak	Adequate	H-1, H-2, H-3, H-4
9		Not Ruptured	No Leak	No Leak	Adequate	No
10		Ruptured	No Leak	No Leak	Inadequate	H-1, H-2, H-3, H-4
11		Not Ruptured	Leak	No Leak	Inadequate	H-2, H-3
12		Not Ruptured	No Leak	Leak	Inadequate	H-2, H-3
13		Ruptured	Leak	No Leak	Inadequate	H-1, H-2, H-3, H-4
14		Not Ruptured	Leak	Leak	Inadequate	H-2, H-3
15		Ruptured	No Leak	Leak	Inadequate	H-1, H-2, H-3, H-4
16		Ruptured	Leak	Leak	Inadequate	H-1, H-2, H-3, H-4

[18] *Other support systems* refers to other systems designed to cool the primary system. This includes the CVCS, SI, CCS, etc. *Adequate* means the system operation is sufficient to provide the cooling normally provided by the SG.

In the case of SG tube rupture, keeping the MSIV open can cause not only equipment damage but also a more immediate shutdown (H-4) via SCRAM and can increase the amount of time the plant will need to remain shut down for repairs. The overfilling of the SG could allow water to enter the steam lines, damaging the delicate turbine pallets and requiring extensive time for repairs. In addition to actual damage, equipment can be overstressed and require more detailed inspections before the plant can be operational again. The additional contamination will also require more time to decontaminate and will result in the generation of more waste. Because keeping the MSIV open during a SG tube rupture will cause a more severe and prolonged shutdown than would otherwise occur with a contained SG tube rupture, H-4 is included in Table 12 for these cases. H-4 is not listed for other cases because it is assumed that keeping the MSIV open after a leak in the main steamline or main feedwater pipe will not cause a more severe or prolonged shutdown than if the MSIV is closed, although it does contribute to the other hazards listed.

Note that for the purpose of reviewing the tables, the rationale behind each of the "hazardous" vs. "not hazardous" decisions should be documented during the analysis. In fact, the context tables can be used to help verify that the necessary rationales and assumptions are documented during the analysis, as opposed to ad-hoc identification of hazardous control actions that may immediately discount and omit non-hazardous control actions entirely. Of course, the non-hazardous rows could easily be omitted from the context tables if desired; however, documenting the conclusions about what behavior is hazardous can be just as important as documenting behavior that is assumed to be non-hazardous. Such documentation may be especially important for other long-term project goals like future change management activities, design re-use in new environments, and other considerations that arise later in the system lifecycle.

A comparison of Table 11 and Table 12 shows that there are conflicts that must be resolved. In both tables, rows 10 to 16 are marked as hazardous. In other words, in these situations it is hazardous to close the MSIV yet hazardous to keep the MSIV open. In some cases, it is possible to revisit the design to eliminate the conflict and provide a safe option. If the conflict cannot be resolved, a decision must be made about what action should be taken in these contexts, that is, which is the *least* hazardous? For this case study, after consultation with nuclear engineers and regulators it was found that rows 10 to 16 may not have been analyzed in previous safety analyses with respect to MSIV control. For the purposes of this research, the consensus was to assume that it may be best to keep the MSIV open in the context of row 10 to maximize the amount of cooling provided even though doing so will contaminate the secondary cooling system and eventually require costly repairs. Rows 11-16, on the other hand, involve leaks in the pipe supplying water to the steam generator and/or the line that carries steam away. If the MSIV is left open in these situations, the amount of water in the steam generator can decrease and eventually lead to less cooling capability or an overcooling transient. Therefore, in these situations (rows 11-16), it was assumed that it may be best to keep the MSIV closed to maximize the amount of cooling provided even though it is only a temporary measure. These solutions were found to differ from current designs of MSIV controllers, which do not act based on the state of other support systems and may automatically close the MSIV during any rupture. Chapter 4 discusses design conflicts in more detail, including search and detection methods that can be performed on STPA results to automatically detect such conflicts.

Both of these assumptions should be reviewed and evaluated carefully by domain experts. The purpose of this research case study was not to provide final solutions to these hazardous situations, but to develop and apply hazard analysis methods that can uncover hazardous control and provide the safety-critical questions that need to be considered. Note that although Table 11 and Table 12 use high-level contexts, the analysis can also be performed in more detail using the techniques described in chapter 5. A more detailed analysis could be necessary if, for example, it is found that the best solution depends on the type of steam generator tube rupture, the amount of pressure in the SG, etc.

Of course, in any of these situations, there are other control actions that need to take place outside the MSIV control loop—they can be analyzed using the same approach. In addition, every effort should be made to prevent many of these contextual conditions from existing in the first place. Although such additional efforts were outside the scope of this initial case study, they are mentioned here to show how the analysis may branch out into other areas of the system to address the issues identified.

3.4 Identifying causal factor scenarios

Although identifying hazardous control actions is an important part of an STPA analysis, it is not enough. As described in chapter 2, once hazardous control actions are identified, STPA Step 2 is performed to identify the potential causes of hazardous control actions and to provide a better understanding of why they might be provided and how they can be prevented. However, accidents can still occur even without hazardous control actions if, for example, correct and safe control actions are provided but not executed or followed by other components in the system. Therefore, STPA Step 2 also identifies the causal factors that can lead to a violation of safety constraints despite safe control actions.

STPA Step 2 has traditionally been performed ad-hoc as a brain-storming exercise based on the generic set of causal factors illustrated in Figure 13. However, if STPA Step 1 is performed using the processes proposed earlier, it is possible to leverage the STPA Step 1 results to help guide users performing STPA Step 2 and to derive some basic application-specific accident scenarios to assist in the brain-storming process.

This section outlines a basic approach to performing STPA Step 2 based on a set of hazardous control actions. Two parts are presented: (1) using the list of hazardous control actions to define scenarios in which safe control actions may not be followed or executed, and (2) using the hazardous control actions to define scenarios that help identify causes of the hazardous control action.

3.4.1 Identifying how safe control actions may not be followed or executed

When hazardous control actions are identified using the structures proposed earlier, they contain information to create accident scenarios in which a provided control action is not hazardous but the action is not followed or executed, leading to a hazard. This is true because the work needed to identify the hazardous control actions also identifies the safety-critical contexts for each command. By using control-theoretic principles, a set of basic accident scenarios can be derived from the STPA Step 1 results. For example, consider the following hazardous control actions from the train example:

Door open command not provided when stopped and emergency

Door open command not provided when stopped at platform and closing on person

Even if the controller correctly provides the door open command in these contexts, some other controller or some downstream component may interfere with the execution of the control action and cause a hazard. From the unsafe control action statements and the safety control structure, the safety-critical behavior of downstream components in the control path can be specified as well as the safety-critical contexts that are important:

Safety-critical command: Open door

Safety-critical function: Door opens when commanded

Safety-critical contexts: When train is stopped at platform; during emergency conditions; and/or while closing on a person

A basic hazardous scenario can be constructed from this information:

Hazardous scenario S-1: Door open command provided but door doesn't open

If this scenario occurs during one of the safety-critical contexts, a hazard will result. Table 13 shows examples of hazardous control actions from STPA Step 1 and the corresponding basic scenarios that can be generated for the case of safe control actions that are not followed.

Table 13: Basic scenarios and safety-critical contexts generated from hazardous control actions

Hazardous control action	**Basic scenario (for control action not followed)**	**Safety-critical contexts**
Door open command not provided when train is stopped at platform and person in doorway	S-1: Door open command provided but door does not open	Train is stopped at platform and person in doorway; train is stopped and emergency exists
Door open command not provided when train is stopped and emergency exists		
Door open command provided when train is moving	S-2: Door open command not provided but door opens	Train is moving; train is stopped unaligned with platform with no emergency
Door open command provided when train is stopped unaligned with platform and there is no emergency		
Door open command is provided more than X seconds after train stops during an emergency	S-3: Door open command provided as required but door only opens X seconds later	Train stopped during an emergency

Using the control structure as a guide, Figure 22 shows how the basic scenario S-1 can be refined into several sub-cases based on the command's control path. The next step is to identify potential causal factors in the highlighted areas of the control loop that would explain the scenario, keeping in mind that each case may have different causes and that some causes may be specific to (or even induced by) the safety-critical contexts identified.

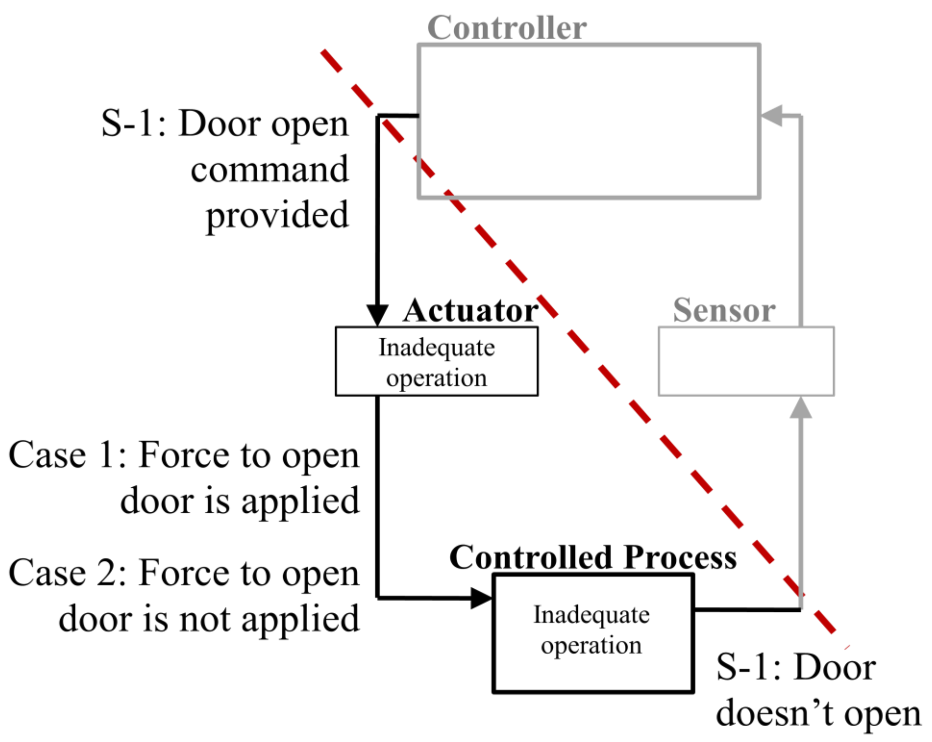

Controller
S-1: Door open command provided
Actuator
Inadequate operation
Sensor
Case 1: Force to open door is applied
Case 2: Force to open door is not applied
Controlled Process
Inadequate operation
S-1: Door doesn't open

Figure 22: Developing an accident scenario using the control structure

For example, the following application-specific causal factors could be identified to explain S-1 and to develop specific design requirements.

Table 14: Examples of causal factors and safety-critical requirements for accident scenario S-1

Application-specific causal factors	**Application-specific design requirements**
Door cannot reverse direction fast enough after closing on a person	Door must designed to facilitate immediate reversal of direction
Emergency conditions make actuator inoperable (e.g. fire)	Actuator must be designed and located to withstand emergency conditions (e.g. fire)
Door jams when closing on an object	Door must be able to withstand closing on a stationary or moving object without jamming

Note that causal factors in this part of the analysis may include potential component failures, but the causal factors are not and should not be limited to component failures. As the examples in Table 14 demonstrate, consideration should also be given to application-specific design flaws and requirements flaws that may explain the basic scenario even when no components have failed.

Although the set of basic scenarios (e.g. S-1, S-2, S-3) can be generated automatically from a list of hazardous control actions and context tables, human input is still required to identify the causal factors in Table 14 that can explain the basic scenarios. However, by making use of parts of the analysis that have already been done, this approach can help reduce the amount of work duplicated between STPA steps 1 and 2 while providing direct traceability between STPA Step 1 context tables and STPA Step 2 scenarios that can be reviewed and verified. In addition, because the scenarios are derived from the context tables, no time is wasted considering scenarios that may not lead to hazards.

Although the examples used to explain the approach in this section are intentionally simplistic, the approach has also been applied successfully to more complex systems. For a more detailed example of scenarios generated from hazardous control actions, see the nuclear power plant system analysis in the appendix.

3.4.2 Identifying the causes of hazardous control actions

Figure 13 describes the generic set of causal factors used in STPA. An important causal factor is a flawed process model: when a controller's belief about the outside world is incorrect or inadequate, it can cause the controller to issue a hazardous control action. The generic set of causal factors in Figure 13 does not provide much information about potential process model flaws in a specific application other than the generic conclusion that the process model variables must match reality. Some additional guidance can be derived from the context tables used in STPA Step 1. In addition to specifying which process model variables are safety-critical, the context tables provide information about the exact contexts in which flaws can and cannot cause hazards. This information can be used to define the application-specific *type* of process model flaw that may cause a hazard. For example, if a train door controller believes the train is at a platform when it is not, then it may cause a hazard by opening the doors. However, the inverse case for the same process variable may not be hazardous—if the door controller believes the train is not aligned with a platform when it actually is, then it may not cause any of the identified hazards.

Information about the exact types of process model flaws that cause hazards is important because it can be used by engineers to design safer sensors, control algorithms, voting logic, or to employ other fault-tolerant and fail-safe techniques.[19] For example, the control algorithms can be designed to assume the train is not at a platform when no input is received from the sensors or when messages are found to be corrupt, and platform sensors can be designed so they indicate no alignment when they fail. This information about hazardous process model flaws can also be used to reduce unnecessary work in the hazard analysis by only focusing on the specific types of flaws that are hazardous rather than all the ways the "train aligned" variable may be incorrect.

Moreover, this kind of information can be derived automatically from the STPA Step 1 context tables, reducing the amount of time an analyst needs to spend in STPA Step 2 thinking about all the ways a process model variable could be flawed and lead to an accident. This technique also provides traceability by allowing the exact hazards relevant to each process model flaw to be readily determined from the existing context tables, which can be helpful when assessing future design changes or re-use applications that alter the process model variables in the design or the hazards involved. Table 15 shows the process model flaws for the train door controller that can be automatically computed from the context tables.

[19] This information could also be used to model the "unknown" [104, 143, 144] values in a SpecTRM model of the system or controller and define control algorithms that make the right decisions when values are unknown.

Table 15: Process model flaws generated from context tables for each hazardous control action

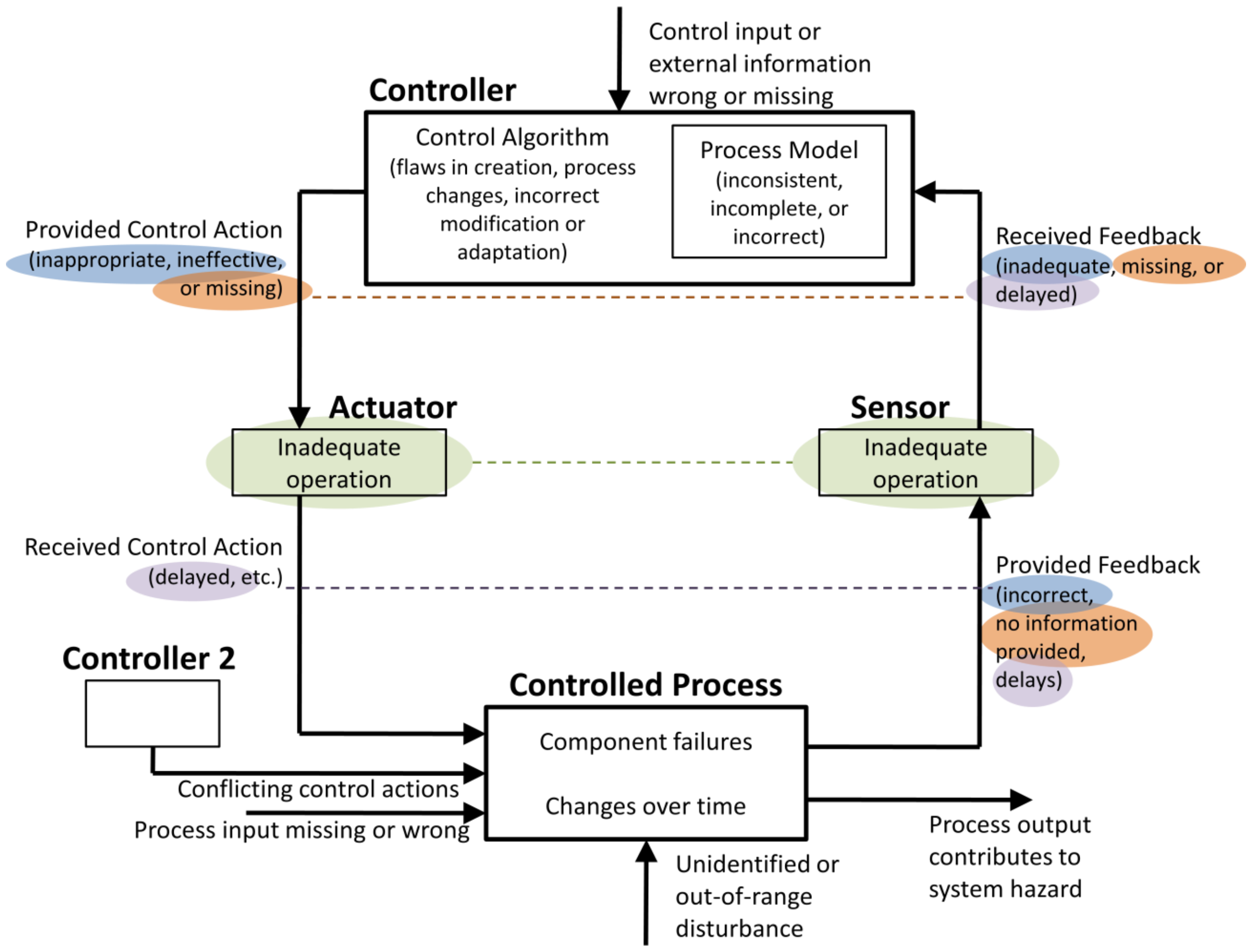
Hazardous control action	**Relevant process model flaws**
Door open command not provided when train is stopped at platform and person in doorway	• Controller incorrectly believes train is moving • Controller incorrectly believes no person in doorway • Controller incorrectly believes train is not aligned
Door open command not provided when train is stopped and emergency exists	• Controller incorrectly believes train is moving • Controller incorrectly believes no emergency exists
Door open command provided when train is moving	• Controller incorrectly believes train is moving
Door open command provided when train is stopped unaligned with platform and there is no emergency	• Controller incorrectly believes train is aligned • Controller incorrectly believes there is an emergency
Door open command is provided more than X seconds after train stops during an emergency	• Delayed realization that train is stopped • Delayed realization of emergency

Notice that in Table 15 there are several hazardous control actions that can be caused by the same process model flaw. For example, "controller believes train is moving" is identified as a cause of three different hazardous control actions. Previous ad-hoc STPA analyses, e.g. [127], can encounter a significant amount of repetition during STPA Step 2 because each hazardous control action is typically analyzed for causes independently of other hazardous control actions. By first generating the relevant process model flaws as in Table 15, the set of unique flaws that may be hazardous in several different ways can be identified thereby reducing repetition and the amount of subsequent analysis effort needed. The unique process model flaws in Table 15 can be automatically generated if the techniques in chapter 4 are used.

Control flaws in the feedback path can cause the controller to issue hazardous control actions by creating process model flaws. As illustrated in Figure 23, the generic set of STPA causal factors contains a substantial amount of symmetry. This suggests that the procedures developed earlier for hazardous control actions and for the control path in general might also be applicable to analyzing the feedback path. In fact, the traditional four types of hazardous control actions defined in STAMP could be translated and applied to the feedback path.

Four types of unsafe feedback:

1. A feedback parameter required for safety is not provided
2. An incorrect feedback parameter is provided that leads to a hazard
3. Correct feedback is provided too late, too early, or out of sequence, causing a hazard
4. Correct continuous feedback is stopped too soon or applied too long, causing a hazard



Control input or external information wrong or missing
Controller
Control Algorithm (flaws in creation, process changes, incorrect modification or adaptation)
Process Model (inconsistent, incomplete, or incorrect)
Provided Control Action (inappropriate, ineffective, or missing)
Received Feedback (inadequate, missing, or delayed)
Actuator
Inadequate operation
Sensor
Inadequate operation
Received Control Action (delayed, etc.)
Provided Feedback (incorrect, no information provided, delays)
Controller 2
Controlled Process
Component failures
Changes over time
Conflicting control actions
Process input missing or wrong
Unidentified or out-of-range disturbance
Process output contributes to system hazard

Figure 23: Symmetry in the generic set of STPA causal factors

Accident scenarios for the feedback path can now be defined in a manner similar to scenarios for the control path. For example, one of the safety-critical process model variables for the train door controller represents whether emergency conditions are present. A basic accident scenario involving the emergency signal can be developed as follows:

Safety-critical condition: Emergency condition present

Safety-critical function: Notify controller of emergency conditions

Hazardous scenario S-4: Emergency condition present but door controller is not notified

Table 16 and Table 17 show basic scenarios that can be generated in this way to explain the process model flaws in Table 15.

Table 16: Basic scenarios for identifying causes of hazardous control actions

Hazardous process model flaws	**Basic scenario (for feedback not provided)**	**Basic scenario (for incorrect feedback provided)**
Controller believes no emergency exists	S-4: Emergency condition exists, but emergency signal not received by controller	S-5: Controller notified of no emergency when emergency exists
Controller believes train is aligned	S-6: Train is unaligned, but unaligned signal is not received by controller	S-7: Aligned signal received by controller when train is unaligned
Controller believes train is moving	S-8: Train is stopped, but stop signal not received by controller	S-9: Moving signal received by controller when train is stopped
Controller believes no person in doorway	S-10: Person in doorway, but controller not notified	S-11: Controller notified doors are clear when person is in doorway
Controller believes there is an emergency	S-12: There is no emergency, but emergency signal is received by controller	S-13: Emergency signal received by controller when there is no emergency
Controller believes train is not aligned	S-14: Train is aligned, but aligned signal is not received by controller	S-15: Unaligned signal received by controller when train is aligned

Table 17: Basic scenarios for identifying timing-related causes of hazardous control actions

Hazardous process model flaws	**Basic scenario (for feedback too late, too early, out of sequence)**
Delayed realization that train is stopped	S-16: Train stops, but controller not notified until after X seconds
Delayed realization of emergency	S-17: Emergency exists, but controller not notified until after X seconds

Similarly to the control path, the accident scenarios for flaws in the feedback can be refined into two sub-cases as shown in Figure 24.The highlighted areas of the control loop can then be analyzed to identify what application-specific causal factors would explain these scenarios. For example, the causal factors in Table 18 could explain S-4 and form the basis for specific design requirements.

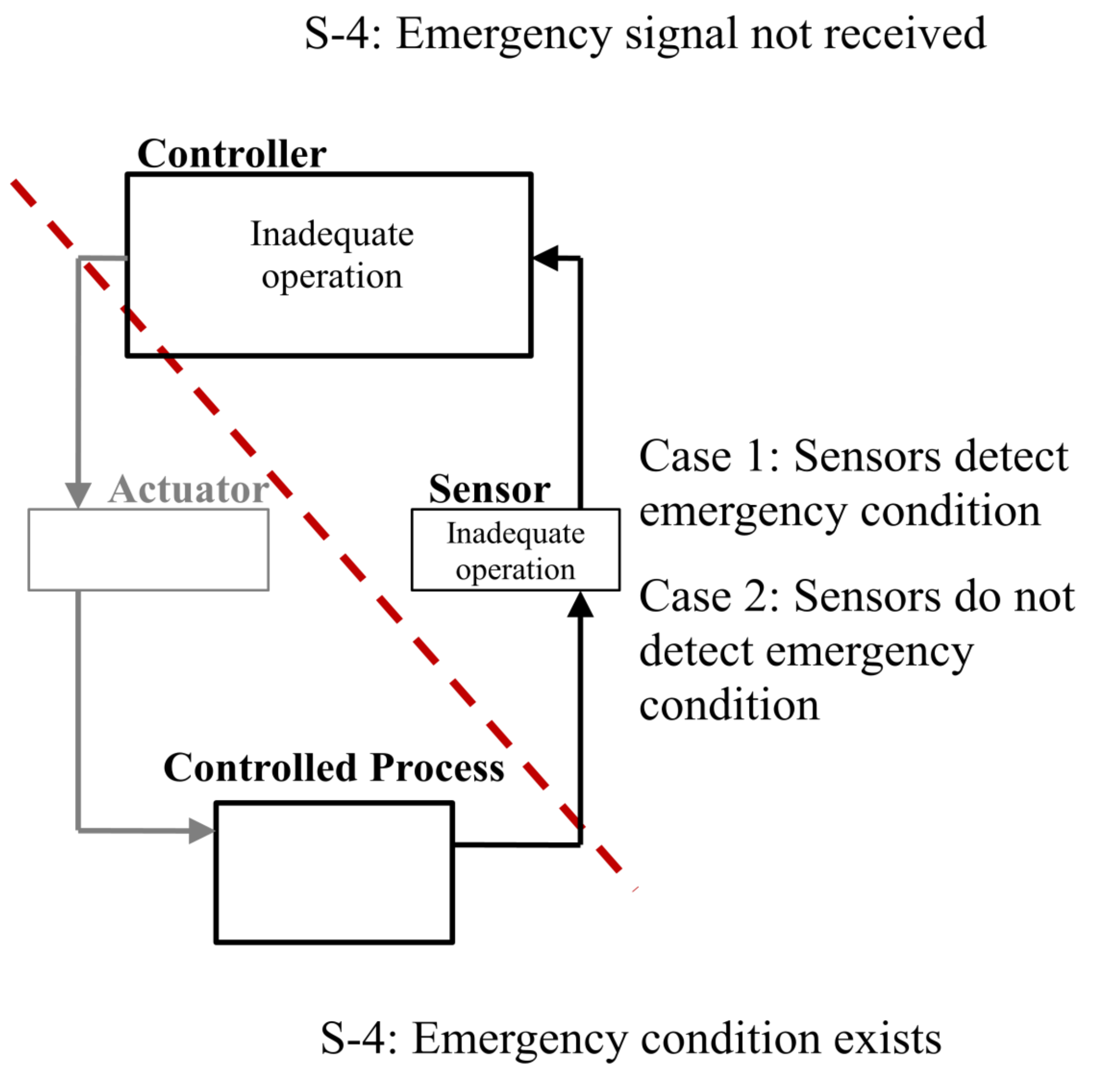

S-4: Emergency signal not received
Controller
Inadequate operation
Actuator
Sensor
Inadequate operation
Case 1: Sensors detect emergency condition
Case 2: Sensors do not detect emergency condition
Controlled Process
S-4: Emergency condition exists

Figure 24: Developing accident scenario S-4 to identify causes of hazardous control actions

Table 18: Examples of causal factors and safety-critical requirements for accident scenario S-4

Application-specific causal factors	**Application-specific design requirements**
Sensor inadequate; hasn't operated for extended periods due to lack of emergency conditions	Sensor must be designed to operate after X years with no detections Sensor must be tested every Y years Sensor must be replaced every Z years
Gas-based sensor overwhelmed by other materials in the environment (e.g. degassing of batteries, fuel, etc.)	Sensor must be designed to operate in the presence of X,Y,Z materials Limit maximum concentration of X,Y,Z in the environment
Wrong type of sensor; sensor operates but doesn't detect emergency condition	Sensors must be designed to detect smoke particles (e.g. ion sensor), photonic indications of flames, etc. Provide manual emergency lever

Although the basic scenarios can be generated from the context tables, human input is still required to identify many of the causal factors that explain the basic scenarios.

As discussed earlier, the examples in this section are intentionally simplified for the purpose of describing the proposed process. For a more detailed example of scenarios generated from hazardous control actions, see the nuclear power plant system analysis in the appendix.

Chapter 4. Formal methods for STPA Hazard Analysis and Requirements Generation

The procedures described in chapter 3 are based on structures that can be formalized and specified mathematically. A formalized STPA can help reason about unsafe behaviors and causal factors in a system and can lead to more advanced methods of performing the hazard analysis efficiently. In addition, formalization would allow automated methods and tools to be defined so that many parts of the hazard analysis can be automatically performed. Other parts of the analysis that utilize engineering creativity and experience can potentially be refined and guided by tools based on the formalization. New techniques can also be developed to detect important types of problems earlier, such as conflicts between multiple safety goals.

Although hazard analysis is used to identify potential causes of hazards and accidents, that is not enough to prevent accidents—a solution needs to be devised and appropriate requirements need to be defined to ensure safe system behavior. However, if a formal STPA hazard analysis is performed, then the results can be specified using a formal language subject to mathematical reasoning and translation into different forms. A formal definition opens the possibility for additional mathematical algorithms to be developed to automatically search for design solutions given the hazard analysis results and produce the set of necessary model-based requirements that enforce safety throughout the system. A formal definition could also facilitate the application of existing formal requirements methods, such as [104, 143, 145], to STPA hazard analysis results and may lead to the eventual integration of these methods into a comprehensive safety-guided design process.

This chapter begins by defining a formal specification for hazardous control actions and develops several automated algorithms based on this specification. Parts of the STPA hazard analysis are shown to be automatable even during early phases of development when system and component models are not available, and new opportunities to assist engineers during a hazard analysis are discussed including the ability to detect design conflicts with safety implications. Algorithms are also developed to automatically generate formal model-based and executable requirements based on STPA results. Finally, these techniques are extended to also consider non-safety-related functional goals of the system, and automated methods to analyze for conflicts in safety vs. non-safety design goals are presented.

4.1 A Formal Specification for Hazardous Control Actions

In this section, a formal specification is introduced and defined for hazardous control actions. This specification is used in later sections to develop automated algorithms that assist in identifying hazardous actions and generating requirements that enforce safe behavior. In addition, although the formal structure is defined here relative to system-level hazards, an identical structure can be applied relative to system-level functions or goals. These parallel structures form the basis for methods later in this chapter that can be used to generate both safety and non-safety-related requirements as well as to detect potential conflicts between the two.

The formal specification is based on the hazardous control action definition developed in chapter 3, as shown in Figure 25 and Figure 26.

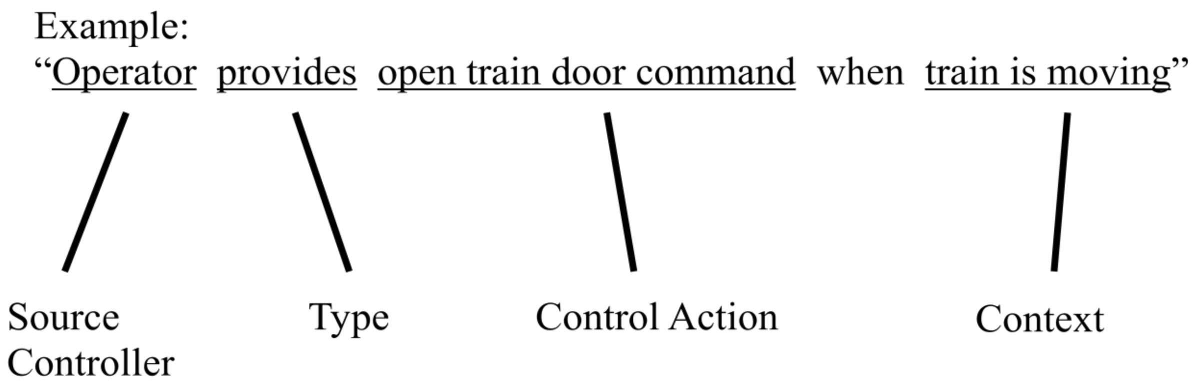

Example:
"Operator provides open train door command when train is moving"
Source Controller
Type
Control Action
Context

Figure 25: Four main elements of a hazardous control action

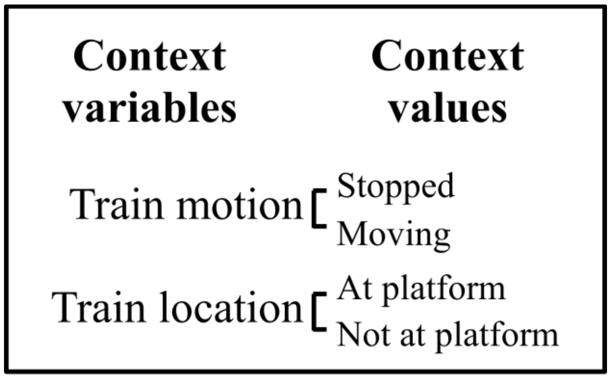


Figure 26: Example of variables and values that compose the context

A hazardous control action in the STAMP accident model can be expressed formally as a four-tuple (SC,T,CA,Co) where:[20]

- SC is the source controller that can issue control actions in the system. The controller may be automated or human.
- T is the type of control action. There are two possible types: *Provided* describes a control action that is issued by the controller while *Not Provided* describes a control action that is not issued.
- CA is the control action (i.e. command) that is output by the controller.
- Co is the context in which the control action is or is not provided.

For example, in the case of an automated train door controller, consider the following hazardous control action: The train door controller provides the *open door* command while the train is moving. This control command can be expressed as (SC,T,CA,Co) where:

SC = Train door controller
T = Provided
CA = Open door command
Co = Train is moving

Each element of a hazardous control action is a member of a larger set, i.e. the following properties must hold:

[20] As described in chapter 3, in some cases a destination field may also be necessary. The structures and syntax in this chapter can be extended to these cases if necessary.

1. SC ∈ $\mathcal{SC}$, where $\mathcal{SC}$ is the set of source controllers in the system
2. T ∈ $\mathcal{T}$, where $\mathcal{T}$ = {Provided, Not Provided}
3. CA ∈ $\mathcal{CA}$(SC), where $\mathcal{CA}$(SC) is the set of control actions that can be provided by controller SC
4. Co ∈ $\mathcal{Co}$(SC), where $\mathcal{Co}$(SC) is the set of potential contexts for controller SC

To assist in enumerating or aggregating individual contexts, the context Co is further decomposed into variables, values, and conditions:

- V is a variable or attribute in the system or environment that may take on two or more values. For example, *train motion* and *train position* are two potential variables for a train.
- VL is a value that can be assumed by a variable. For example, *stopped* is a value that can be assumed by the variable *train motion*.
- Cd is a condition expressed as a single variable/value pair. For example, *train motion is stopped* is a condition.
- The context Co is the combination of one or more conditions and defines a unique state of the system or environment in which a control action may be given.

The following additional properties related to the context of a hazardous control action can therefore be defined:

5. V ∈ $\mathcal{V}$(SC), where $\mathcal{V}$(SC) is the set of variables relevant to the system hazards $\mathcal{H}$
6. VL ∈ $\mathcal{VL}$(V), where $\mathcal{VL}$(V) is the set of values that can be assumed by variable V
7. Cd = (V, VL) ∈ $\mathcal{Cd}$(SC), where $\mathcal{Cd}$(SC) is the possible set of conditions for controller SC
8. Co = (Co_1, Co_2, ...), where each Co_i is independent. That is, no two Co_i refer to the same variable V.

Finally, each hazardous control action must be linked to a system-level hazard:

9. To qualify as a hazardous control action, the action (SC,T,CA,Co) must be able to cause a hazard H ∈ $\mathcal{H}$, where $\mathcal{H}$ is the set of system level hazards.

A hazardous control action expressed as a four-tuple (SC,T,CA,Co) must satisfy the above properties 1-9.

4.2 Automatable method to generate hazardous control actions

This section defines a formal method that can be used to automate much of the manual process described in the previous section. Based on the formal structure defined in section 4.1, a set of potentially hazardous control actions can be enumerated given certain information about the system. The information needed is:

- $\mathcal{H}$: the set of system-level hazards
- Ş$\mathcal{C}$: the set of source controllers in the system
- $\mathcal{CA}$(SC): the set of control actions for each controller SC
- $\mathcal{V}$: the set of variables relevant to the hazards $\mathcal{H}$
- $\mathcal{VL}$(V): the set of potential values for each variable V

Most, if not all, of this information can be determined well in advance of the detailed design of a system. The set $\mathcal{H}$ is typically determined during the Preliminary Hazard Analysis (PHA) of the system. The set Ş$\mathcal{C}$ can be extracted from a preliminary control structure of the system. (SC) can be defined from the basic control channels in the control structure. The set $\mathcal{V}$ is identical to the process model variables in the control structure, and can be derived from the set of hazards $\mathcal{H}$. The potential values (V) are also found in the process model, and can be defined once $\mathcal{V}$ is known.

Given this basic information about the system, properties 1-8 from section 4.1 can be applied to automatically generate a list of potential hazardous control actions in the form of combinations of (SC,T,CA,Co). First, a controller SC is selected from the set Ş$\mathcal{C}$. Then the set of conditions (SC) is generated by pairing each variable in $\mathcal{V}$ with each value in $\mathcal{VL}$(V). Then the set of contexts (SC) is generated by combining each independent condition from $\mathcal{Cd}$(SC). Finally, the list of potentially hazardous control actions for the selected controller SC is generated using each element of $\mathcal{T}$, (SC), and $\mathcal{Co}$(SC) to produce a series of unique combinations. This process can be repeated for each controller SC in the set Ş$\mathcal{C}$.

This process guarantees that properties 1-8 from section 4.1 are satisfied. If a formal behavioral model of the system exists for every component, and it is complete and accurate with respect to all safety implications, then automated methods can be used to trim this set of potentially hazardous control actions to only keep those that satisfy property 9. However, because a detailed behavioral model of the system typically does not exist during the earliest phases of development, and because models for non-software components are usually incomplete and can only approximate physical reality, it may not be possible to automatically apply property 9 in practice. Instead of trying to analyze an existing formal model, which has been studied extensively, the proposal here is intended to be applicable to early development stages before any formal system model exists. Here, STPA is used to help engineers iteratively *define* the necessary system behavior/model based on high-level hazards and objectives. As a result, the model can be derived in such a way that it will be safe. This safety-driven design approach is more direct and often more efficient than waiting for a detailed formal model, analyzing it, and then performing rework or adding patches and protective systems as discussed in chapter 1.

Instead of applying property 9 automatically, this final step can be performed by the engineering team during early development. Because the algorithm above generates combinations that satisfy all other criteria, the generated list is a superset of the actual hazardous control actions. Therefore this task is essentially a trimming exercise: the team does not need to add any new hazardous control actions—they only need to remove non-hazardous control actions from the list based on their engineering knowledge, skills, and experience. In fact, this can be done by engineers without any training in the formal logic described above.

Finally, for each potential hazardous control action that is provided (T = Provided), timing information such as potentially hazardous delays within a given context must be considered. For example, suppose it is not hazardous to provide a door open command while the train is stopped and there is an emergency. In fact, this behavior may be exactly what is expected of the system. However, providing the door open command *too late* in that context could certainly be hazardous even if the control action is eventually provided. This condition can be addressed by adding the columns *hazardous if provided too early* and *hazardous if provided too late* as described in chapter 3. This timing information can be incorporated into the context element of a hazardous control action, or it can be translated directly into a set of timing constraints.

Experience using this approach on real systems such as spacecraft [129] and the air transportation system [127, 146] has led to the identification of safety-critical requirements that were never considered during the normal development of these systems. Other examples can be found in the appendix, which includes context tables for nuclear power plant control systems that were generated using the algorithms in this section. In addition, the example in chapter 5 presents more detailed hazardous control actions that can be generated using this approach along with algorithms and techniques to address scalability to complex systems.

4.3 Automated generation of model-based requirement specifications

Identifying the hazardous behaviors to avoid is necessary, but it is not enough: specific requirements need to be created to define the actual behavior necessary to prevent hazards and existing requirements need to be checked to verify that these hazardous behaviors will not occur. Because hazardous control actions have been defined with a formal representation, it is possible to compare these actions against an existing formal model-based specification to determine whether the hazardous control actions are precluded. Furthermore, if no formal specification exists, it is possible to automatically generate the parts of the specification necessary to ensure hazardous behavior is prevented.

The following functions can be defined from the set of hazardous control actions:

- HP(H, SC, CA, Co): This function is *True* if and only if hazard H results from controller SC providing command CA in context Co. This function is defined for all H ∈ $\boldsymbol{\mathcal{H}}$, SC ∈ Ş$\boldsymbol{\mathcal{C}}$, CA ∈ (SC) , Co ∈ $\boldsymbol{\mathcal{Co}}$(SC).
- HNP(H, SC, CA, Co): This function is *True* if and only if hazard H results from controller SC not providing command CA in context Co. This function is defined for all H ∈ $\boldsymbol{\mathcal{H}}$, SC ∈ Ş$\boldsymbol{\mathcal{C}}$, CA ∈ (SC) , Co ∈ $\boldsymbol{\mathcal{Co}}$(SC).

The formal specifications and control algorithms to be generated can be expressed as the following function:

- R(SC, CA, Co): This function is *True* if and only if controller SC is required to provide command CA in context Co. This function must be defined for all SC ∈ Ş$\boldsymbol{\mathcal{C}}$, CA ∈ (SC), Co ∈ $\boldsymbol{\mathcal{Co}}$(SC).

The function R must satisfy certain criteria to prevent hazardous behavior. Firstly, any control action that is hazardous in a given context must not be provided by the control algorithm in that context:

$$\forall\ H \in \mathcal{H},\ SC \in \mathcal{SC},\ CA \in \mathcal{CA}(SC),\ Co \in \mathcal{Co}(SC):\ HP(H, SC, CA, Co) \Rightarrow \neg R(SC, CA, Co) \qquad (1)$$

In addition, if a control action that is absent in a given context will produce a hazard, then the control action must be provided by the control algorithm in that context:

$$\forall\ H \in \mathcal{H},\ SC \in \mathcal{SC},\ CA \in \mathcal{CA}(SC),\ Co \in \mathcal{Co}(SC):\ HNP(H, SC, CA, Co) \Rightarrow R(SC, CA, Co) \qquad (2)$$

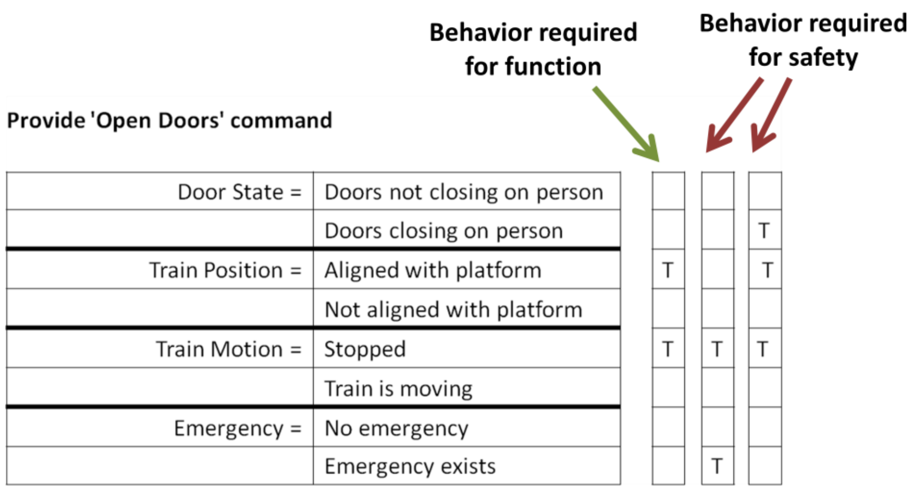
The required behavior R can then be generated by searching the set of possible (SC, CA, Co) and assigning values that satisfy these two criteria. Any behavior appearing in HNP must appear in R, and any behavior that appears in HP must be absent from R. If the same behavior appears in HNP and HP, then no R can satisfy both criteria[21].

The resulting requirements in R can be specified using a formal requirements language. In this section, a formal requirements language called Specification Tools and Requirements Methodology Requirements Language (SpecTRM-RL) [104] is used. SpecTRM-RL is a blackbox formal system modeling language that uses a state-based representation of a system. In addition to mathematical constructs, SpecTRM-RL provides a graphical representation of formal requirements that can be used effectively by engineers and developers with very little explanation. Although the examples in this chapter are expressed using SpecTRM-RL graphical tables, almost any formal state-based requirements language could be adapted and used.

Figure 27 shows the SpecTRM-RL representation of R generated automatically using a software tool that implements the procedure above. The example is based on the context tables for the train door controller discussed in chapter 3. Each row in the table describes a state or input to the controller and a possible value for that state or input. The three columns on the right side describes the AND-OR logic used to determine whether the command should be provided. OR relationships exist between columns while AND relationships exist between rows. Empty cells are treated as

[21] For more information, see the discussion of conflicts and design flaws in the next section.

"don't cares" or wildcards. For example, the first column specifies that if the train is stopped and aligned with a platform, then the door open command must be provided. The middle column specifies another situation in which the door open command must be provided: if the train is stopped and an emergency exists. The two right-most columns define the required behavior for safe operation and represent the requirements in R. The left column was not generated from the context tables in chapter 3 because the context tables describe only hazardous control actions, not non-hazardous control actions required for safety. However, this column can be generated automatically using the approach in section 4.5, which extends context tables to non-safety-related functional requirements.


Behavior required for function

Behavior required for safety

Provide 'Open Doors' command

Door State =	Doors not closing on person			
	Doors closing on person			T
Train Position =	Aligned with platform	T		T
	Not aligned with platform			
Train Motion =	Stopped	T	T	T
	Train is moving			
Emergency =	No emergency			
	Emergency exists		T	

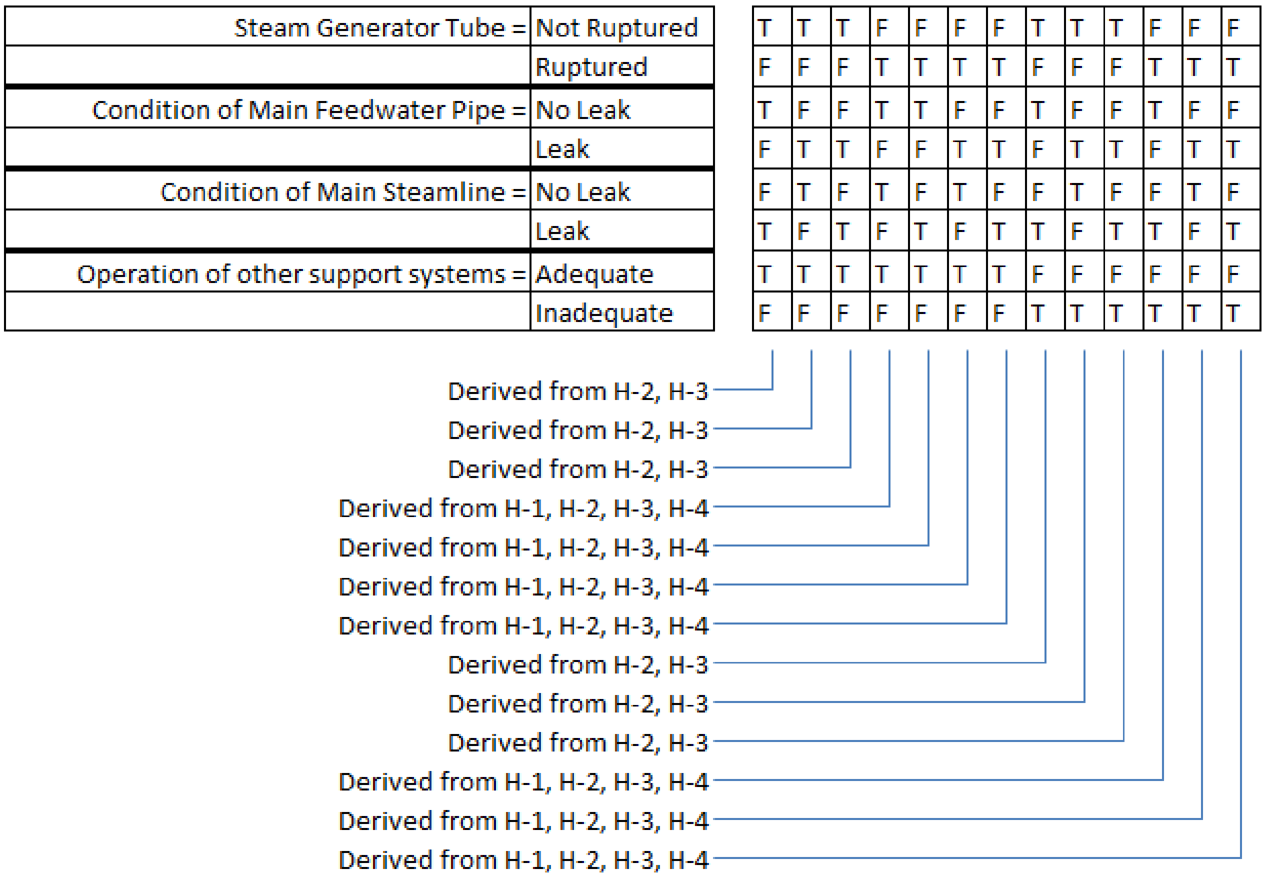
Figure 27: Example SpecTRM-RL table for the door open command

Although this example is intentionally simple for purpose of demonstrating the procedure[22], the same approach can be applied to more complex control systems to evaluate existing requirements or to generate the initial set of requirements. For example, Figure 28 below shows the SpecTRM-RL representation of the high-level requirements in R as generated for the nuclear reactor power plant system in chapter 3.[23] Chapter 5 describes an example of more detailed lower level requirements for the same system.

[22] Although not shown in the table, SpecTRM models also capture timing information as described in [104, 143, 144]. The necessary timing information can be derived from the STPA Step 1 context table columns "too early" and "too late" or from the defined contexts.

[23] Note that conflicts first had been resolved, which was done following the procedure in the next section.

Provide 'Close MSIV valve' command



Steam Generator Tube =	Not Ruptured	T	T	T	F	F	F	F	T	T	T	F	F	F
	Ruptured	F	F	F	T	T	T	T	F	F	F	T	T	T
Condition of Main Feedwater Pipe =	No Leak	T	F	F	T	T	F	F	T	F	F	T	F	F
	Leak	F	T	T	F	F	T	T	F	T	T	F	T	T
Condition of Main Steamline =	No Leak	F	T	F	T	F	T	F	F	T	F	F	T	F
	Leak	T	F	T	F	T	F	T	T	F	T	T	F	T
Operation of other support systems =	Adequate	T	T	T	T	T	T	T	F	F	F	F	F	F
	Inadequate	F	F	F	F	F	F	F	T	T	T	T	T	T

Derived from H-2, H-3
Derived from H-2, H-3
Derived from H-2, H-3
Derived from H-1, H-2, H-3, H-4
Derived from H-1, H-2, H-3, H-4
Derived from H-1, H-2, H-3, H-4
Derived from H-1, H-2, H-3, H-4
Derived from H-2, H-3
Derived from H-2, H-3
Derived from H-2, H-3
Derived from H-1, H-2, H-3, H-4
Derived from H-1, H-2, H-3, H-4
Derived from H-1, H-2, H-3, H-4

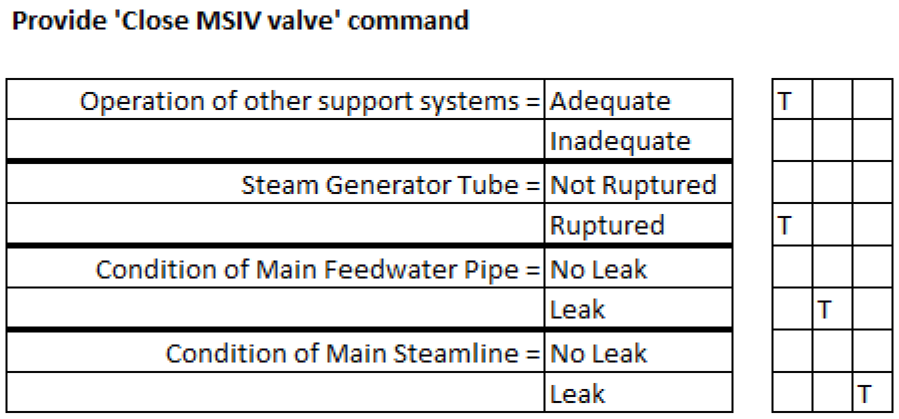
Figure 28: SpecTRM-RL requirements with traceability to system hazards for the nuclear power plant system studied in chapter 3

The requirements in Figure 28 are complete in that they include all contexts in which the MSIV valve command must be provided, and each context can be refined to include more detail if necessary. Figure 28 also shows the traceability between each column and the corresponding system hazards that can occur if that behavior is not enforced. The columns in Figure 28 do not use “don’t cares” or wildcards. They represent the disjunctive normal form of the formal requirement:

$$R(\text{“MSIV Controller”}, \text{“Close MSIV”}, Co) = Co \wedge \neg SGTR \wedge \neg MFPL \wedge MSL \wedge \neg SSI \vee Co \wedge \neg SGTR \wedge MFPL \wedge \neg MSL \wedge \neg SSA \vee Co \wedge \neg SGTR \wedge MFPL \wedge MSL \wedge \neg SSA \vee Co \wedge SGTR \wedge \neg MFPL \wedge \neg MSL \wedge \neg SSA \vee Co \wedge SGTR \wedge \neg MFPL \wedge MSL \wedge \neg SSA \vee Co \wedge SGTR \wedge MFPL \wedge \neg MSL \wedge \neg SSA \vee Co \wedge SGTR \wedge MFPL \wedge MSL \wedge \neg SSA \vee Co \wedge \neg SGTR \wedge \neg MFPL \wedge MSL \wedge SSA \vee Co \wedge \neg SGTR \wedge MFPL \wedge \neg MSL \wedge SSA \vee Co \wedge \neg SGTR \wedge MFPL \wedge MSL \wedge SSA \vee Co \wedge SGTR \wedge \neg MFPL \wedge MSL \wedge SSA \vee Co \wedge SGTR \wedge MFPL \wedge \neg MSL \wedge SSA \vee Co \wedge SGTR \wedge MFPL \wedge MSL \wedge SSA$$

where	SGTR=T*rue* iff Steam Generator Tube is Ruptured,
	MFPL=*True* iff Condition of Main Feedwater Pipe is Leak
	MSL=*True* iff Condition of Main Steamline is Leak
	SSI=*True* iff Operation of other support systems is Inadequate

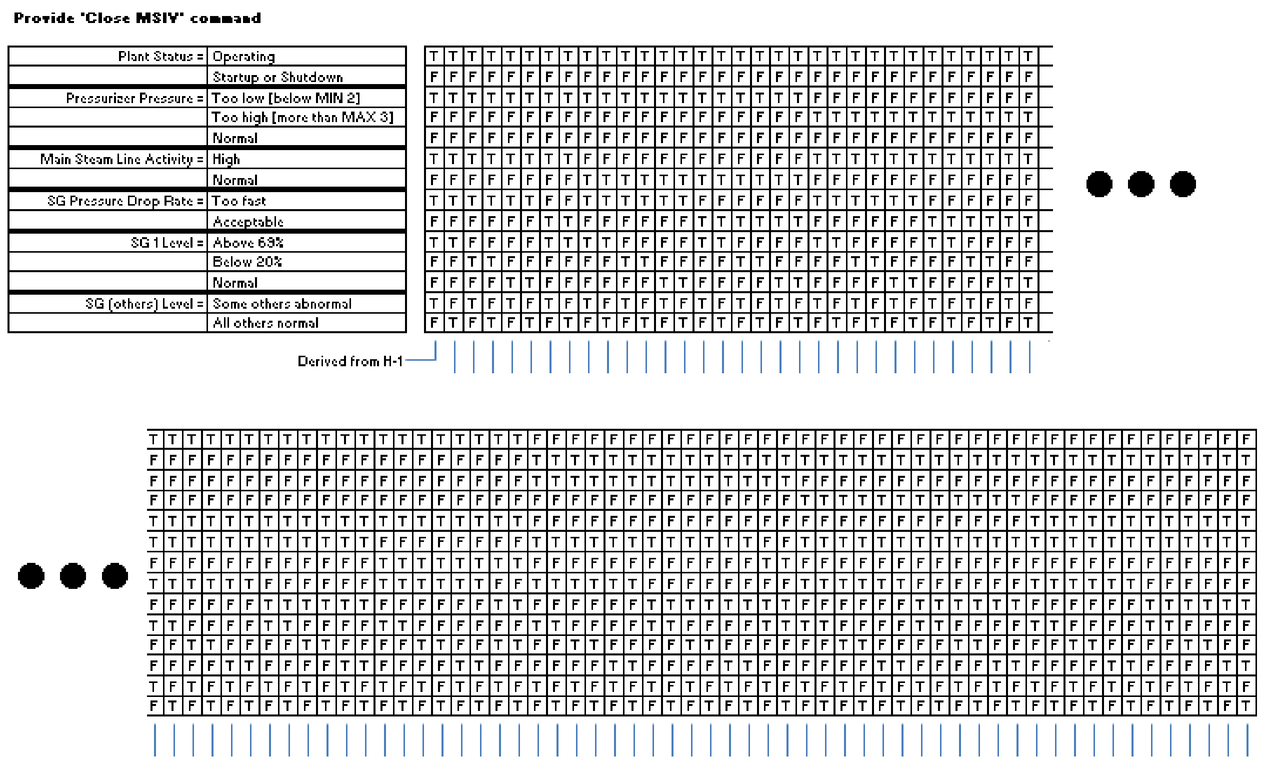
Automated algorithms can be used to produce a simpler representation of the same requirement for engineering review if desired, as shown in Figure 29. Note that this requirement is based on the assumptions discussed earlier when the context tables were created. Namely, it was assumed that if there is a steam generator tube rupture and other support systems are inadequate, then it may be best to keep the MSIV valve open to maximize cooling capability until the temperature and reactivity can be controlled by other means. As mentioned earlier, these assumptions should be reviewed carefully by domain experts. The goal here is to show how requirements can be automatically generated given the hazard analysis and assumptions—not to provide the final word about dealing with steam generator tube and other leaks.


Provide 'Close MSIV valve' command

Operation of other support systems =	Adequate	T		
	Inadequate			
Steam Generator Tube =	Not Ruptured			
	Ruptured	T		
Condition of Main Feedwater Pipe =	No Leak			
	Leak		T	
Condition of Main Steamline =	No Leak			
	Leak			T

Figure 29: SpecTRM-RL requirements for the nuclear power plant system studied in chapter 3

Lower-level requirements can also be generated using this approach. For example Figure 30 below shows the formal requirements for the same controller at a more detailed level. However, the extensions described in chapter 5 were found to be a much more efficient and practical way of producing these lower-level requirements as the complexity increases.



Provide 'Close MSIY' command

Plant Status =	Operating	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
	Startup or Shutdown	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
Pressurizer Pressure =	Too low [below MIN 2]	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	F	F	F	F
	Too high [more than MAX 3]	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	T	T	T	T
	Normal	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
Main Steam Line Activity =	High	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	T	T	T	T
	Normal	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	F	F	F	F
SG Pressure Drop Rate =	Too fast	T	T	T	T	T	T	F	F	T	T	T	T	T	T	F	F	F	F	F	F	T	T	T	T	T	T	F	F	F	F	F	F
	Acceptable	F	F	F	F	F	F	T	T	F	F	F	F	F	F	T	T	T	T	T	T	F	F	F	F	F	F	T	T	T	T	T	T
SG 1 Level =	Above 63%	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F	F
	Below 20%	F	F	T	T	F	F	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F
	Normal	F	F	F	F	T	T	F	F	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F	F	T	T
SG (others) Level =	Some others abnormal	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F
	All others normal	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T

Derived from H-1

● ● ●

● ● ●

T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	F	F	F	F
T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	T	T	T	T
T	T	T	T	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	T	T	T	T	F	F	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
F	F	F	F	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	F	F	F	F	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
T	T	T	T	T	T	F	F	F	F	F	F	T	T	T	T	T	T	F	F	T	T	T	T	T	T	F	F	F	F	F	F	F	F	T	T	T	T	T	T	F	F	F	F	F	F	T	T	T	T	T	T	F	F	F	F	F	F
F	F	F	F	F	F	T	T	T	T	T	T	F	F	F	F	F	F	T	T	F	F	F	F	F	F	T	T	T	T	T	T	T	T	F	F	F	F	F	F	T	T	T	T	T	T	F	F	F	F	F	F	T	T	T	T	T	T
T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F	F
F	F	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	T	T	F	F	T	T	F	F	F	F	T	T	F	F	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F
F	F	F	F	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F	F	T	T	F	F	F	F	T	T
T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F
F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T

Figure 30: Low-level SpecTRM-RL requirements generated for the nuclear power plant system

4.4 Identifying conflicts and potential design flaws

As mentioned in the previous section, if there is no solution to equations (1) and (2), then there exists a context for which there is no safe action—i.e. it is hazardous for the controller to provide the control action while at the same time it is hazardous for the controller not to provide the control action. If the same system hazard results from both action and inaction, then there is a fundamental design flaw that must be resolved. If action causes one hazard while inaction causes another hazard, then there is a conflict in the design between two safety-related goals that must be resolved. Although in some cases the hazards can be prioritized and conflicts can be resolved by accepting the lesser hazard while preventing the greater hazard, in general it is often necessary and possible to find a design solution that does not cause any hazards.

The following additional criterion can be defined to search the hazard analysis context tables or generated requirements and detect these problems as early as possible:

$$\forall\ H_1 \in \boldsymbol{\mathcal{H}}, H_2 \in \boldsymbol{\mathcal{H}}, SC \in \boldsymbol{\mathcal{S}}, CA \in \boldsymbol{\mathcal{CA}}(SC), Co \in \boldsymbol{\mathcal{Co}}(SC)\text{: } HP(H_1, SC, CA, Co) \Rightarrow \neg HNP(H_2, SC, CA, Co) \tag{3}$$

The third criterion above is a consistency check that can be applied to the hazardous control actions independently of the formal specification R. If this criterion does not hold, there is a design or requirements flaw in the system. Both action and inaction by controller SC would lead to a hazard and violate a safety requirement.

In a limited set of cases, the identified conflicts can be automatically resolved. If the set of hazards are prioritized and the conflict is between hazards of different priorities, then the "best" control action (i.e. the one that causes the least important hazard) can be automatically selected to resolve the conflict. However, in general it is not acceptable to create a design that intentionally causes hazards. The best solutions usually involve fundamental design changes that can only be identified through careful application of engineering creativity and expertise. Although these conflicts cannot usually be automatically resolved, they can be automatically detected and flagged for review by the engineering team.

4.5 Extensions for non-safety-related functional requirements

The first column in Figure 27, representing the non-safety-related functional requirement for the train door controller, can be generated automatically using a procedure similar to the one developed for safety requirements. Although the context tables were defined relative to system-level hazards that must be prevented, context tables can be developed for system-level functional goals in the same way. By deriving the functional behavior of the system in the same way that hazardous behavior was derived, automated algorithms can be used to generate the full set of safety and non-safety requirements as the example in Figure 27 shows. Functional specifications can then be generated along with the safety-related specifications by following a parallel method.

In addition to HP and HNP, which capture hazardous control actions, a new function FP can be introduced to define functional control actions—control actions that are needed to achieve functional goals:

- FP(F, SC, CA, Co): This function is *True* if and only if system-level function F must be achieved, in whole or in part, by controller SC providing command CA in context Co

The function FP can be defined by identifying which control actions in each context are necessary to achieve the system-level functions $\mathcal{F}$. The same process used in chapter 3 to identify hazardous control actions with context tables can be used, with the exception that system-level functions $\mathcal{F}$ are considered instead of the system-level hazards $\mathcal{H}$. The required behavior R can then be computed as in the previous section, but with one additional criterion to capture the functional behavior:

$$\forall\ F \in \mathcal{F},\ S \in \c{S}\mathcal{C},\ CA \in \mathcal{CA}(SC),\ Co \in \mathcal{Co}(SC):\ FP(F, SC, CA, Co) \Rightarrow R(SC, CA, Co) \qquad (4)$$

Applying this criterion, any behavior appearing in FP must also appear in R. Note that if the same behavior appears in FP and HP, then there is a design or requirements flaw in the system because the same control action is both necessary to achieve a system-level function yet prohibited because it presents a system-level hazard. In this case, no R exists that prevents the hazards while achieving the system functions. This new type of conflict can be identified with the following criterion:

$$\forall\ H \in \mathcal{H},\ F \in \mathcal{F},\ SC \in \c{S}\mathcal{C},\ CA \in (SC),\ Co \in \mathcal{Co}(SC):\ HP(H, SC, CA, Co) \Rightarrow \neg FP(F, SC, CA, Co) \qquad (5)$$

This final criterion describes a consistency check that can be implemented to detect conflicts between hazardous and functional behavior. Similarly to conflicts between different hazards, these conflicts usually cannot be automatically resolved but they can be automatically detected and flagged for review by the engineering team.

Chapter 5. Scalability to complex systems

Like any hazard analysis technique, the approaches described in this dissertation can require more effort as the complexity of the system being analyzed increases. The basic problem of ensuring safety in complex systems—a property that emerges from the combined behavior of many components with many interactive effects—increases significantly as complexity introduces more and more potentially hazardous or dysfunctional interactions among components. Although it may be unavoidable that more complex systems require more careful analyses, there are a number of ways to manage system complexity during the analysis and frame the problem in ways that are easier for engineers and experts to comprehend and reason about. This chapter presents a number of techniques that can be used to deal with system complexity and improve the scalability of the methods proposed in this dissertation.

5.1 Abstraction and hierarchy

The first technique—*abstraction and hierarchy*—is commonly used to help people deal with complexity. STPA is a top-down approach and makes extensive use of abstraction and hierarchy in developing and analyzing hazards, safety constraints, and control structures. Abstraction can also be applied to control actions to allow high-level analyses that can later be refined. Problems that are solved at high levels of abstraction may not need to be analyzed at lower levels of analysis, thereby reducing the total analysis effort. For example, when analyzing new aviation procedures for pilots, the control action "pilots execute passing maneuver" can be analyzed to identify problems and solutions at that high level as opposed to first considering the various lower level control actions—like entering information into autopilot systems or communicating with copilots—that together make up the passing maneuver. This control action abstraction can significantly reduce the number of context tables that need to be created to complete STPA Step 1 and to begin identifying new procedures, requirements, and causal factors in STPA Step 2.

More important, the use of abstraction is essential in defining the columns for the context tables. For example, the train door example in chapter 3 used one column labeled "emergency". Clearly there are many different kinds of emergencies that could occur—fire, smoke, toxic gases, etc. The context table could be created with separate columns for each of these cases, however the table would quickly grow and become much more complex than is necessary at this stage. For the purpose of determining high-level door controller behavior, the exact type of emergency is not what matters; what matters is that an evacuation is required. Therefore, “emergency” is defined in chapter 3 as any condition that requires passenger evacuation. Further analysis can and should eventually identify and define all the types of emergencies that might require evacuation so that design efforts can be made to prevent those occurrences. However, the analysis of the door controller—including the context tables—can be performed at a higher level of abstraction before that level of detail is defined.

In fact, an important goal of the approaches in this dissertation is to provide benefit to early phases of design when very little is known about the system. In these cases, abstraction is natural because most details have not yet been defined, and the analysis can be used to drive the design and determine which details may need to be defined or developed first.

The process model hierarchy in chapter 3 is another example of employing abstraction and hierarchy. Instead of trying to identify all the low-level process variables from the start, the process model hierarchy provides a way to deal with complex processes by deriving progressively lower levels of detail from the system-level conditions in the defined hazards. The process model hierarchy can also provide a conduit for connecting the high-level analysis with lower levels of refinement, allowing traceability between different levels of analysis and providing the ability to select appropriate and feasible levels of analysis for complex systems. In fact, section 5.5 describes how the process model hierarchy can be exploited to automatically generate detailed low-level context tables given a smaller set of abstract context tables along with a few pieces of additional information.

5.2 Logical simplification

The second technique—*logical simplification*—was already employed when introducing the context tables for the train door as seen in Table 19. In this example, the four columns of variables each with two possible values would require a 16 row table using a brute force approach. However, Table 19 only requires five rows. By reducing similar rows with “doesn’t matter” terms, the table can be drastically simplified. For example, the last row in Table 19 represents eight unique contexts.

This simplification is possible because if the train is moving, then the specific value of the other variables don't matter – keeping the door closed is not hazardous.

Table 19: Context table for the *open door* control action

Control Action	Train Motion	Emergency	Train Position	Door State	Hazardous if not provided in this context?
Door open command not provided	Train is stopped	No emergency	Aligned with platform	Person not in doorway	No[24]
Door open command not provided	Train is stopped	No emergency	Not aligned with platform	Person not in doorway	No
Door open command not provided	Train is stopped	No emergency	Aligned with platform	Person in doorway	Yes
Door open command not provided	Train is stopped	No emergency	Not aligned with platform	Person in doorway	No[25]
Door open command not provided	Train is stopped	Emergency exists	(doesn't matter)	(doesn't matter)	Yes
Door open command not provided	Train is moving	(doesn't matter)	(doesn't matter)	(doesn't matter)	No

Automated tools can help perform this reduction automatically or assist the user in identifying and specifying these simplifications, as discussed in section 5.6.

5.3 Continuous process model variables

The context table examples provided in this dissertation describe a number of continuous process variables. For example, train motion is a continuous variable with an infinite number of possible values. However, it is not necessary to consider an infinite number of values or even a large number of values. What is important for the purpose of analyzing door commands that cause a system

[24] This row is not hazardous because it does not lead to any of the system-level hazards (see H-1,H-2,H-3 in the previous section). If the hazards and accidents included in the safety analysis were extended to include inconvenience to the passengers, then this row would describe a hazardous control action.

[25] For the purpose of this analysis it is assumed that in this case it is best to keep the door closed and alert a crew member to assist the potentially trapped passenger.

hazard is simply whether the train is stopped (velocity equals zero) or moving (velocity not equal to zero). Through careful discretization of process variables up front based on the system hazards, the complexity of the context tables and subsequent analysis can be significantly reduced. Although this process is dependent on the user, the process model hierarchy and system hazards can provide guidance to help reduce unnecessary complexity in the analysis. In addition, automated tools discussed in section 5.6 can provide ways to easily expand or simplify the defined values during the analysis as necessary (e.g. to split "train is moving" into "train is moving slow" and "train is moving fast"). The tools in section 5.6 can also automatically identify whether the set of values in a finished context table can be further reduced, which can significantly simplify subsequent steps in the hazard analysis.

It is important to note that the set of values defined for each variable does not necessarily need to be detailed, but they must be complete so that every possibility is included. For example, the set *train is moving* and *train is stopped* is complete because the set includes every possibility. Analyzing the set of values—even at high levels of abstraction during early development stages—can lead to important insights. For example, the set *door open* and *door closed* may appear complete at first, but upon closer inspection the continuous nature of the variable can immediately reveal a potentially critical state—*partially open*—that must be accounted for in the analysis.

5.4 Defining rules to quickly create and evaluate large tables

Although the first three techniques can be particularly useful during early stages of development, it is also possible to work with larger and more detailed context tables during later stages of development. Although most of the context table can be generated automatically given information in the control structure, as discussed in chapter 4 the final column must still be defined manually in most cases. When faced with this task, it can be more efficient to define a set of rules such that automated tools can fill out the table. For example, in the nuclear reactor example of Table 20, a rule can be defined to represent the fact that closing the MSIV any time support systems are inadequate will risk causing H-2 (reactor temperature too high). The automated tool can then apply the rule to generate rows 9-16 of column 6 with the appropriate hazard. Similar rules can be defined from basic principles to quickly fill the entire table.

Table 20: Context table for *Operator provides Close MSIV* control action

	1	2	3	4	5	6	7	8
	Control Action	**Steam Generator Tube**	**Condition of Main Feedwater Pipe**	**Condition of Main Steamline**	**Operation of other support systems**	**Control Action Hazardous?**	**Control Action Hazardous if Too Late?**	**Control Action Hazardous if Too Early?**
1	*Close MSIV*	Not Ruptured	No Leak	No Leak	Adequate	H-4	H-4	H-4
2		Ruptured	No Leak	No Leak	Adequate	No	H-1, H-2, H-3, H-4	H-3, H-4
3		Not Ruptured	Leak	No Leak	Adequate	No	H-2, H-3, H-4	No
4		Not Ruptured	No Leak	Leak	Adequate	No	H-2, H-3, H-4	No
5		Ruptured	Leak	No Leak	Adequate	No	H-1, H-2, H-3, H-4	H-3, H-4
6		Not Ruptured	Leak	Leak	Adequate	No	H-2, H-3, H-4	No
7		Ruptured	No Leak	Leak	Adequate	No	H-1, H-2, H-3, H-4	H-3, H-4
8		Ruptured	Leak	Leak	Adequate	No	H-1, H-2, H-3, H-4	H-3, H-4
9		Not Ruptured	No Leak	No Leak	Inadequate	H-2, H-4	H-2, H-4	H-2, H-4
10		Ruptured	No Leak	No Leak	Inadequate	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4
11		Not Ruptured	Leak	No Leak	Inadequate	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4
12		Not Ruptured	No Leak	Leak	Inadequate	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4
13		Ruptured	Leak	No Leak	Inadequate	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4
14		Not Ruptured	Leak	Leak	Inadequate	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4
15		Ruptured	No Leak	Leak	Inadequate	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4
16		Ruptured	Leak	Leak	Inadequate	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4

Although this approach is similar to logical simplification, it does not result in a simplified table and a different process is used—each rule is defined for a specific hazard and the rules are applied sequentially as the table is developed. The rule-based approach applies only to a completely enumerated set of rows—each of which are mutually exclusive and collectively exhaustive—and can produce much more complex tables while requiring less effort than a brute force approach. One advantage is that overlapping rules can be quickly defined from basic principles without first considering if all cases are covered or whether rules may present conflicts. Once the rules are defined, automated methods can then generate the table, apply logical simplification, detect whether overlapping rules conflict, and detect whether there are any rows for which no rules apply (indicating an incomplete set of rules).

Although concepts in this dissertation are primarily intended to help guide early stages of development, this rule-based approach has been used successfully to define tables with hundreds of rows using only a few well-understood and easy to evaluate rules. Larger tables are also possible, although the technique in the next section is a much more practical way to include such low levels of detail.

5.5 Automatically generating low-level tables

Because STPA is a top-down approach, higher levels of behavior are analyzed before more detailed lower levels of behavior. It should be possible, therefore, to leverage information and analysis that has already been performed at higher levels to derive lower-level context tables. This section describes a technique that can be used to generate extremely detailed context tables from more abstract tables and information.

5.5.1 Train door controller example

First consider the high-level context table for the train door controller, reproduced in Table 21. This context table defines the effect of a control action (hazardous, nonhazardous) given variables in the first level of the process model hierarchy (train motion, emergency, etc.). Although lower-level tables could be defined by repeating the whole process with lower level process model variables, doing so can be tedious and inefficient because it does not leverage the information already in this table. What kind of information is needed in addition to Table 21 to define the same table at a lower level of detail? The new information needed is the precise relationship between the first and second levels of variables in the process model hierarchy.

Table 21: Context table for the *open door* control action

Control Action	**Train Motion**	**Emergency**	**Train Position**	**Door State**	**Hazardous if not provided in this context?**
Door open command not provided	Train is stopped	No emergency	Aligned with platform	Person not in doorway	No[26]
Door open command not provided	Train is stopped	No emergency	Not aligned with platform	Person not in doorway	No
Door open command not provided	Train is stopped	No emergency	Aligned with platform	Person in doorway	Yes
Door open command not provided	Train is stopped	No emergency	Not aligned with platform	Person in doorway	No[27]
Door open command not provided	Train is stopped	Emergency exists	(doesn't matter)	(doesn't matter)	Yes
Door open command not provided	Train is moving	(doesn't matter)	(doesn't matter)	(doesn't matter)	No

An example process model hierarchy for the train door controller can be constructed as follows:

[26] This row is not hazardous because it does not lead to any of the system-level hazards (see H-1,H-2,H-3 in the previous section). If the hazards and accidents included in the safety analysis were extended to include inconvenience to the passengers, then this row would describe a hazardous control action.

[27] For the purpose of this analysis it is assumed that in this case it is best to keep the door closed and alert a crew member to assist the potentially trapped passenger.

Example process model hierarchy for train door controller:

- Door obstructed {obstructed, not obstructed}
    - Light curtain reading {blocked, not blocked}
    - Door force sensor reading {normal, door pushed open}
- Train motion {moving, stopped}
    - Speed sensor #1 status {continuous speed}
    - Speed sensor #2 status {continuous speed}
    - Speed sensor #3 status {continuous speed}
- Train platform alignment {aligned, not aligned}
    - Left platform sensor {aligned, not aligned}
    - Right platform sensor {aligned, not aligned}
- Emergency {no emergency, evacuation required}
    - Fire present {normal, fire detected}
        - Engine compartment fire sensor {normal, fire detected}
        - Passenger compartment fire sensor {normal, fire detected}
    - Smoke present {normal, smoke detected}
        - Ionization smoke sensor {normal, smoke detected}
        - Optical smoke sensor {normal, smoke detected}
    - Toxic gas sensor {normal, toxic gas detected}

To define the precise relationship between the first and second levels of process model variables, the SpecTRM-RL tables in Figure 31 could be defined.

Door_obstructed inferred to be ....

		obstructed when:		*not obstructed* when:
Light curtain =	Blocked	T		
	Not Blocked			T
Door force sensor =	Normal			T
	Door pushed open		T	

Train_motion inferred to be ....

		stopped when:	*moving* when:		
Speed sensor #1 status =	Stopped	T			
	Moving		T		
Speed sensor #2 status =	Stopped	T			
	Moving			T	
Speed sensor #3 status =	Stopped	T			
	Moving				T

Train_platform_alignment inferred to be ....

		not aligned when:		*aligned* when:
Left platform sensor =	Aligned			T
	Not Aligned	T		
Right platform sensor =	Aligned			T
	Not Aligned		T	

Emergency inferred to be ....

		no emergency when:	*evacuation required* when:		
Fire present =	Normal	T			
	Fire detected		T		
Smoke present =	Normal	T			
	Smoke detected			T	
Toxic gas sensor =	Normal	T			
	Toxic gas detected				T

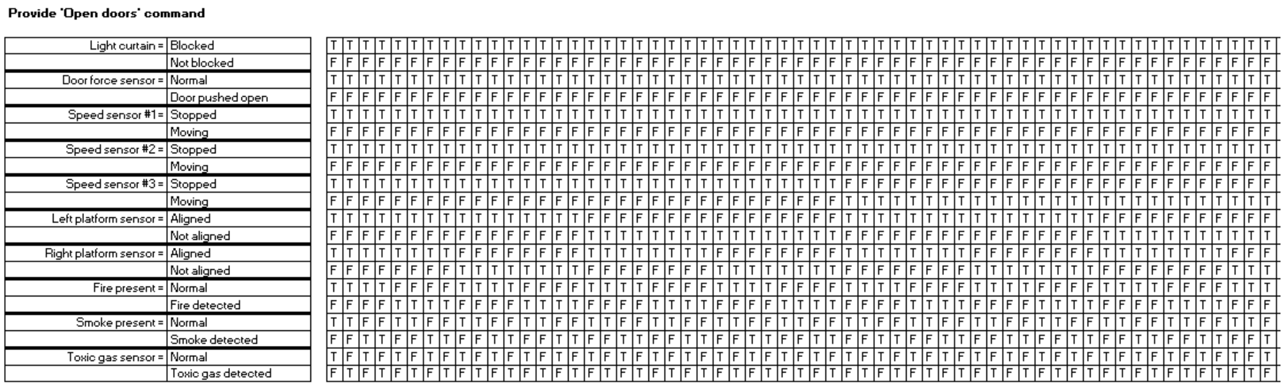
Figure 31: Example SpecTRM-RL tables defining the relationships between process model variables

From this basic information, more detailed context tables can be automatically generated by substituting each process model variable in the high-level context table with the set of lower level process model variables defined in Figure 31. Table 22 shows the first part of the automatically generated low-level context table for the train door controller. The table is quite large and only part can be reproduced here. Although it would be unreasonable to ask engineers to read this table and perform analysis on it, a formal black-box model of the system can be constructed from this information using automated techniques and reduced into a form that can be understood and evaluated.

Table 22: Partial low-level generated context table for train door controller

Light curtain	**Door force sensor**	**Speed sensor #1**	**Speed sensor #2**	**Speed sensor #3**	**Left platform sensor**	**Right platform sensor**	**Fire present**	**Smoke present**	**Toxic gas sensor**	**Hazardous if not provided?**
Blocked	Normal	Stopped	Stopped	Stopped	Not aligned	Aligned	Fire detected	Normal	Normal	Yes
Blocked	Normal	Stopped	Stopped	Stopped	Not aligned	Aligned	Fire detected	Normal	Toxic gas detected	Yes
Blocked	Normal	Stopped	Stopped	Stopped	Not aligned	Aligned	Fire detected	Smoke detected	Normal	Yes
Blocked	Normal	Stopped	Stopped	Stopped	Not aligned	Aligned	Fire detected	Smoke detected	Toxic gas detected	Yes
Blocked	Normal	Stopped	Stopped	Stopped	Not aligned	Not aligned	Normal	Normal	Normal	No
Blocked	Normal	Stopped	Stopped	Stopped	Not aligned	Not aligned	Normal	Normal	Toxic gas detected	Yes
Blocked	Normal	Stopped	Stopped	Stopped	Not aligned	Not aligned	Normal	Smoke detected	Normal	Yes
Blocked	Normal	Stopped	Stopped	Stopped	Not aligned	Not aligned	Normal	Smoke detected	Toxic gas detected	Yes
Blocked	Normal	Stopped	Stopped	Stopped	Not aligned	Not aligned	Fire detected	Normal	Normal	Yes
Blocked	Normal	Stopped	Stopped	Stopped	Not aligned	Not aligned	Fire detected	Normal	Toxic gas detected	Yes
Blocked	Normal	Stopped	Stopped	Stopped	Not aligned	Not aligned	Fire detected	Smoke detected	Normal	Yes
Blocked	Normal	Stopped	Stopped	Stopped	Not aligned	Not aligned	Fire detected	Smoke detected	Toxic gas detected	Yes
Blocked	Normal	Stopped	Stopped	Moving	Aligned	Aligned	Normal	Normal	Normal	No
Blocked	Normal	Stopped	Stopped	Moving	Aligned	Aligned	Normal	Normal	Toxic gas detected	No
Blocked	Normal	Stopped	Stopped	Moving	Aligned	Aligned	Normal	Smoke detected	Normal	No
Blocked	Normal	Stopped	Stopped	Moving	Aligned	Aligned	Normal	Smoke detected	Toxic gas detected	No
…	…	…	…	…	…	…	…	…	…	…

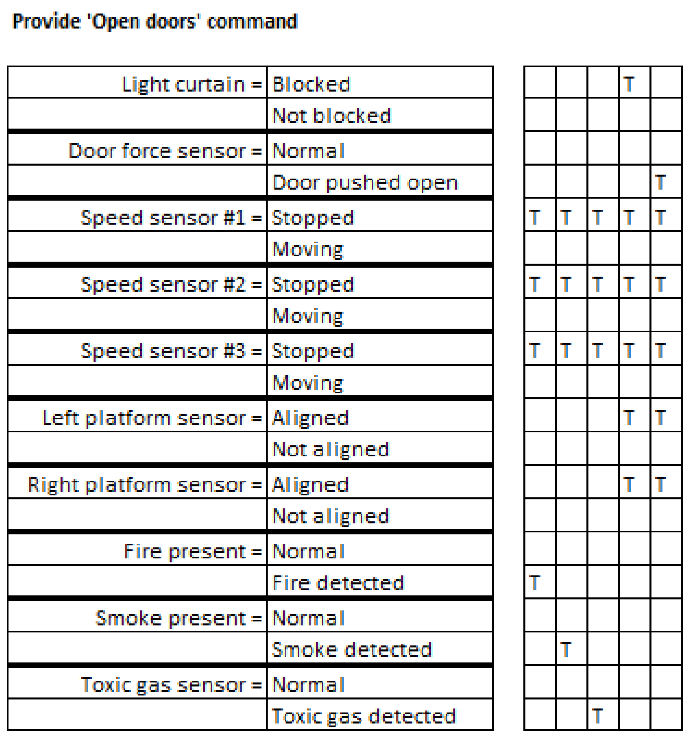
Detailed requirements and control algorithms can be generated from the low-level context tables using the automated methods in chapter 4. Figure 32 shows the first part of the SpecTRM-RL requirements generated from the low-level context tables for the train door controller. The full generated SpecTRM-RL table is quite large; however, applying logical simplification techniques based on the formal semantics defined in chapter 4, the specification can be automatically reduced to an equivalent but much smaller form. Figure 33 shows the logically simplified final SpecTRM-RL table that is generated based on the high-level context tables (e.g. Table 21) and defined relationships between process model variables (e.g. Figure 31).



Provide 'Open doors' command

Light curtain =	Blocked	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
	Not blocked	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
Door force sensor =	Normal	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
	Door pushed open	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
Speed sensor #1 =	Stopped	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
	Moving	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
Speed sensor #2 =	Stopped	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
	Moving	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
Speed sensor #3 =	Stopped	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
	Moving	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
Left platform sensor =	Aligned	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	F	F	F
	Not aligned	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	T	T	T
Right platform sensor =	Aligned	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	F	F	F
	Not aligned	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	T	T	T
Fire present =	Normal	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T
	Fire detected	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F
Smoke present =	Normal	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F
	Smoke detected	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T
Toxic gas sensor =	Normal	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T
	Toxic gas detected	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F

Figure 32: Partial low-level SpecTRM-RL table generated for the train door controller example



Provide 'Open doors' command

Light curtain =	Blocked				T	
	Not blocked					
Door force sensor =	Normal					
	Door pushed open					T
Speed sensor #1 =	Stopped	T	T	T	T	T
	Moving					
Speed sensor #2 =	Stopped	T	T	T	T	T
	Moving					
Speed sensor #3 =	Stopped	T	T	T	T	T
	Moving					
Left platform sensor =	Aligned				T	T
	Not aligned					
Right platform sensor =	Aligned				T	T
	Not aligned					
Fire present =	Normal					
	Fire detected	T				
Smoke present =	Normal					
	Smoke detected		T			
Toxic gas sensor =	Normal					
	Toxic gas detected			T		

Figure 33: Logically simplified low-level SpecTRM-RL table generated for the train door controller example

5.5.2 Nuclear power plant example

The same process can be applied to more complex systems, such as the nuclear power plant system studied in chapter 3 and in the appendix. The high-level context tables in chapter 3 are defined in terms of high-level contexts such as whether or not there is a steam generator tube rupture. However, in reality the software controllers do not receive information at this high level; they must infer the condition of the steam generator tube, for example, from a set of lower-level inputs including various pressure sensors, water levels, and radioactivity sensors. Therefore, lower level requirements are needed to specify the black-box behavior of the controller in terms of actual inputs and outputs.

In other words, high-level context tables in chapter 3 produce requirements about the desired controller output in terms of the controller's internal process model as shown in (1) of Figure 34 below. Another step is required to infer the state of process model variables from the various inputs such as pressure readings, water levels, etc. as indicated by (2) in the figure. This inference process can be flawed leading to hazardous control actions, and such flaws may only be identified through a low-level analysis. Therefore, the software requirements ultimately need to define the black-box requirements of the whole controller as indicated by (3) in the figure. Once (1) and (2) are specified at a high level using relatively simple context tables and SpecTRM-RL tables respectively, the detailed requirements for (3) can be automatically generated.

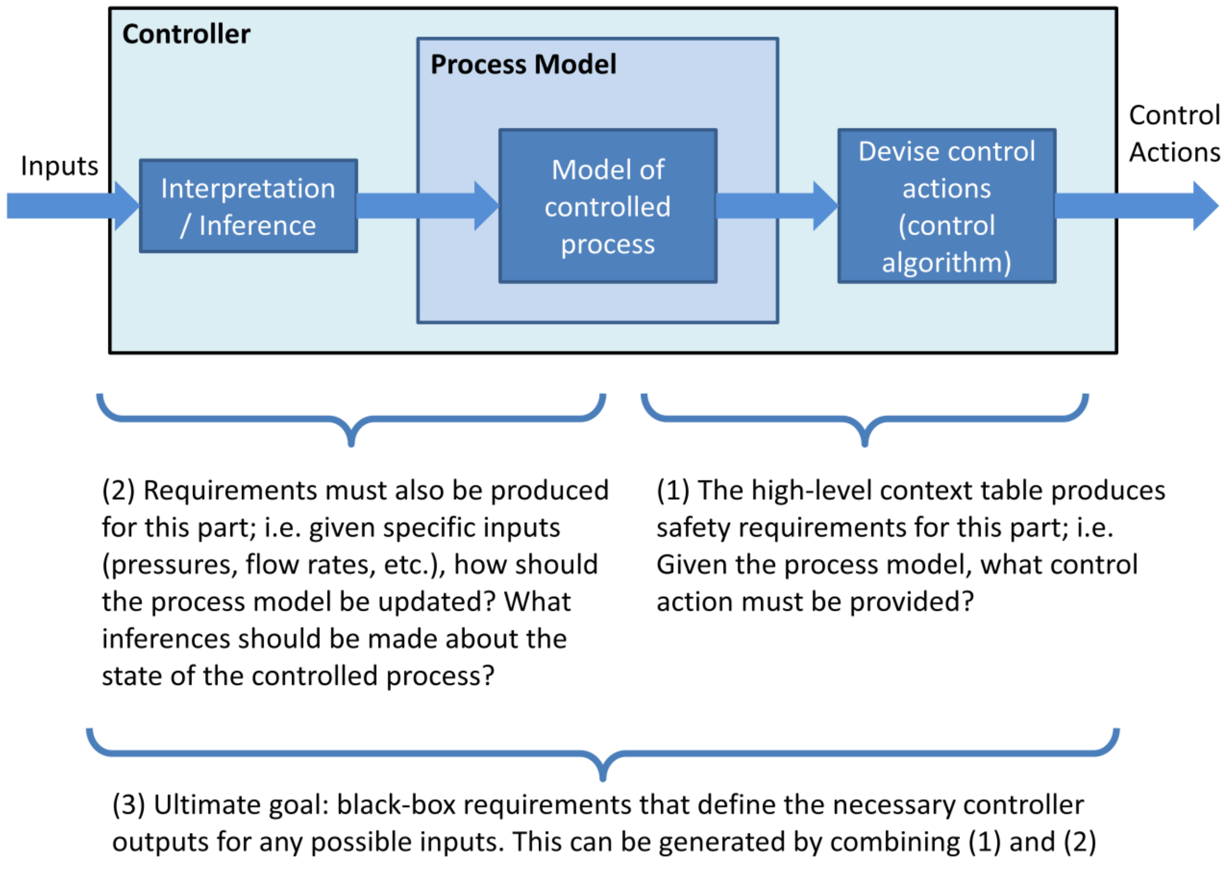

Controller
Process Model
Inputs
Interpretation / Inference
Model of controlled process
Devise control actions (control algorithm)
Control Actions
(2) Requirements must also be produced for this part; i.e. given specific inputs (pressures, flow rates, etc.), how should the process model be updated? What inferences should be made about the state of the controlled process?
(1) The high-level context table produces safety requirements for this part; i.e. Given the process model, what control action must be provided?
(3) Ultimate goal: black-box requirements that define the necessary controller outputs for any possible inputs. This can be generated by combining (1) and (2)

Figure 34: Application of high-level and low-level requirements to a controller

Figure 35 shows an example of SpecTRM-RL tables that could be used to define the relationship between controller inputs and updates to the internal process model variables. For example, how would the Protection System controller know when there is a Steam Generator Tube Rupture (SGTR)? The PS has inputs indicating the SG Water Level and the radioactivity of the SG water. An SGTR would have the effect of raising the SG Water Level and contaminating the SG water with radioactive coolant. Therefore, the controller could infer the existence of an SGTR when the water level is too high or radioactivity is detected.

Condition of Steam Generator Tube inferred to be .... ***ruptured*** when: ***not ruptured*** when:

		ruptured		*not ruptured*
Radioactivity sensor =	Normal			T
	Radioactive	T		
Steam generator water level =	Too low			
	Normal			
	Too high		T	F

Condition of Main Feedwater Pipe inferred to be .... ***leaking*** when: ***not leaking*** when:

		leaking			*not leaking*
Steam generator pressure drop rate =	More than X	T			F
	Less than X				
Steam generator pressure =	More than Y				
	Less than Y		T		F
Containment pressure =	More than Z			T	F
	Less than Z				

Condition of Main Steam Line inferred to be .... ***leaking*** when: ***not leaking*** when:

		leaking			*not leaking*
Steam generator pressure drop rate =	More than X	T			F
	Less than X				
Steam generator pressure =	More than Y				
	Less than Y		T		F
Containment pressure =	More than Z			T	F
	Less than Z				

Operation of other support systems inferred to be .... ***adequate*** when: ***not adequate*** when:

		adequate			*not adequate*
Safety injection system =	Operating	T			F
	Not operating				
Emergency feedwater system =	Operating		T		F
	Not operating				
Emergency cooling system =	Operating			T	F
	Not operating				

Figure 35: Example SpecTRM-RL tables defining the relationships between process model variables for the nuclear power plant example.[28]

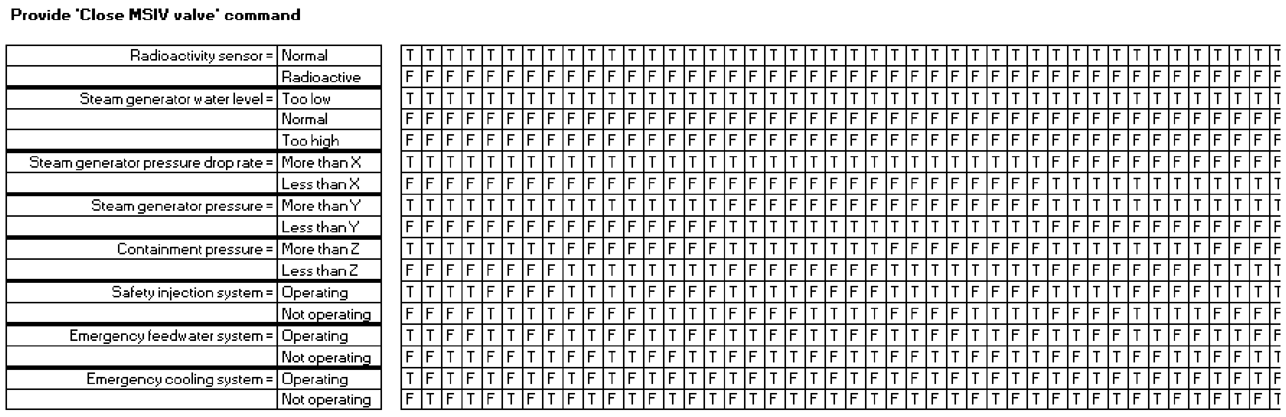
[28] Note that these are only examples. The goal here is to demonstrate how low-level details could be incorporated into a comprehensive hazard analysis. The accuracy of these tables will depend on the specific design considered and the assumptions made. Although not possible with the limited funding provided by the Nuclear Regulatory Commission for

Once the high-level context tables and the process model update algorithms have been defined, the low-level context tables can be generated as shown in Table 23 below. Using the methods defined in chapter 4, the black-box requirements can then be automatically generated from the low-level context table as shown in Figure 36. The generated requirements can then be logically simplified and displayed as SpecTRM-RL black-box requirements as shown in Figure 37. Although it is not practical to review incredibly large tables such as Table 23 and Figure 36, the logically equivalent requirements generated in Figure 37 can be easily understood and reviewed by domain experts.

this research case study, in practice these relationships and assumptions need to be carefully reviewed by domain experts before they are used to generate black-box requirements and models.

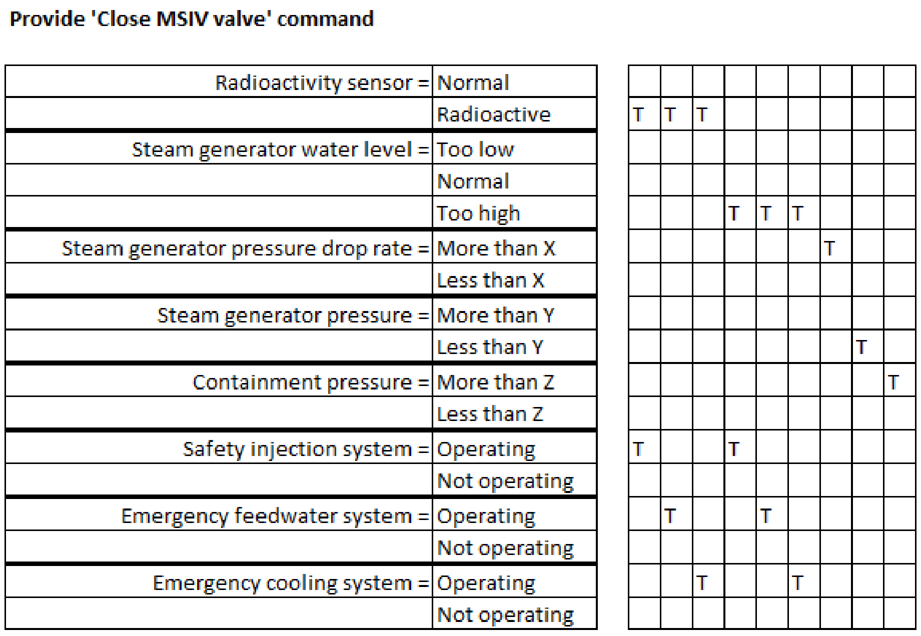
Table 23: Partial low-level generated context table for the nuclear reactor example

Radioactivity sensor	**Steam generator water level**	**Steam generator pressure drop rate**	**Steam generator pressure**	**Containment pressure**	**Safety injection system**	**Emergency feedwater system**	**Emergency cooling system**	**Hazardous to not close MSIV? (to keep MSIV open)**
Normal	Too low	More than X	Less than Y	Less than Z	Not operating	Operating	Operating	Yes
Normal	Too low	More than X	Less than Y	Less than Z	Not operating	Operating	Not operating	Yes
Normal	Too low	More than X	Less than Y	Less than Z	Not operating	Not operating	Operating	Yes
Normal	Too low	More than X	Less than Y	Less than Z	Not operating	Not operating	Not operating	Yes
Normal	Too low	Less than X	More than Y	More than Z	Operating	Operating	Operating	Yes
Normal	Too low	Less than X	More than Y	More than Z	Operating	Operating	Not operating	Yes
Normal	Too low	Less than X	More than Y	More than Z	Operating	Not operating	Operating	Yes
Normal	Too low	Less than X	More than Y	More than Z	Operating	Not operating	Not operating	Yes
Normal	Too low	Less than X	More than Y	More than Z	Not operating	Operating	Operating	Yes
Normal	Too low	Less than X	More than Y	More than Z	Not operating	Operating	Not operating	Yes
Normal	Too low	Less than X	More than Y	More than Z	Not operating	Not operating	Operating	Yes
Normal	Too low	Less than X	More than Y	More than Z	Not operating	Not operating	Not operating	Yes
Normal	Too low	Less than X	More than Y	Less than Z	Operating	Operating	Operating	No
Normal	Too low	Less than X	More than Y	Less than Z	Operating	Operating	Not operating	No
Normal	Too low	Less than X	More than Y	Less than Z	Operating	Not operating	Operating	No
Normal	Too low	Less than X	More than Y	Less than Z	Operating	Not operating	Not operating	No
…	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**


Provide 'Close MSIV valve' command

Radioactivity sensor =	Normal	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
	Radioactive	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
Steam generator water level =	Too low	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
	Normal	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
	Too high	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
Steam generator pressure drop rate =	More than X	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	F	F	F	F
	Less than X	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	T	T	T	T
Steam generator pressure =	More than Y	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	T	T	T	T
	Less than Y	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	F	F	F	F
Containment pressure =	More than Z	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	F	F	F	F
	Less than Z	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	T	T	T	T
Safety injection system =	Operating	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T
	Not operating	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F
Emergency feedwater system =	Operating	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F
	Not operating	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T
Emergency cooling system =	Operating	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F
	Not operating	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T

Figure 36: Partial low-level SpecTRM-RL table generated for the nuclear reactor example


Provide 'Close MSIV valve' command

Radioactivity sensor =	Normal									
	Radioactive	T	T	T						
Steam generator water level =	Too low									
	Normal									
	Too high				T	T	T			
Steam generator pressure drop rate =	More than X							T		
	Less than X									
Steam generator pressure =	More than Y									
	Less than Y								T	
Containment pressure =	More than Z									T
	Less than Z									
Safety injection system =	Operating	T			T					
	Not operating									
Emergency feedwater system =	Operating		T			T				
	Not operating									
Emergency cooling system =	Operating			T			T			
	Not operating									

Figure 37: Logically simplified low-level SpecTRM-RL table generated for the nuclear reactor example

5.6 Automated tools

The techniques described above offer several opportunities for the development of automated tools to assist users performing the analysis. Because the methods are based on formal structures, even parts of the analysis that cannot be automated can still benefit from tools that can restructure the problem in new ways and perform user-directed low-level tasks to improve efficiency, reduce repetition, and leverage results from earlier parts of the analysis. This section describes a number of automated tools, some of which are currently being developed.

Given an existing context table, automated tools can help with logical simplification by identifying the areas that can be simplified to reduce the size of the table. For incomplete tables being developed, tools can assist the user in identifying and specifying these simplifications. For example, a user could highlight multiple rows and ask the tool to expand or reduce the set of contexts by inserting or removing “doesn’t matter” cells.

Tools can also help users create and modify the process model variables. For example, if it is discovered that the train door controller behavior depends on whether the train is moving forward or backward, tools could allow the user to select a “train is moving” cell and split it into two sub-cases. Another possibility is to help users understand how important a process model variable is, for example, by identifying which hazards could result from a specific process model flaw or which process model variables have no affect and can be removed from columns in the context table. Tools could help users understand which process model variables are the most important by prioritizing them based on the severity of the hazards that each process model flaw can lead to. The process model values can also be analyzed to determine whether the values for a given variable can be further simplified or reduced without losing information in the context table. For example, if the set of values for a process model variable includes (high, normal, low), then tools can analyze the context table to automatically determine whether a smaller set such as (high, not high) contains all the necessary information relevant to that table.

A promising tool currently in development automatically applies a set of rules to generate larger context tables. The tool allows users to specify any number of rules and can detect when rules conflict with each other or when the set of rules is incomplete. The tool can also be used to quickly modify existing tables, for example, to reflect design changes or controller re-use in new systems and environments.

Finally, tools can help users define the process model hierarchy and the relationship between levels in the hierarchy, permitting automatic generation of low-level context tables and detailed requirements. The generated requirements could then be represented in SpecTRM-RL and executed or imported into a requirements or systems engineering framework such as Intent Specifications

[147] and existing software tools like SpecTRM [144] that help document traceability and document rationale behind decisions. Existing specification analysis techniques, such as those defined in [143], can then be applied to assess consistency and completeness criteria.

[Page intentionally left blank]

Chapter 6. Conclusions and future work

This dissertation presented a number of extensions to STPA hazard analysis that provide a framework for discovering potentially hazardous behavior and the causal factors behind accidents in complex human- and software-intensive systems. In chapter 3 it was found that a general structure can be defined for hazardous control actions, and a more rigorous procedure was developed to systematically identify hazardous control actions based on that general structure. The STAMP concept of a process model was extended to create a process model hierarchy that provides traceability between system hazards and the contextual elements of a hazardous control action. Once established, the process model hierarchy can also guide subsequent iterations of the STPA analysis by defining relationships between varying levels of the analysis.

Chapter 3 also developed a method of forming basic accident scenarios to guide the identification of causal factors in an STPA analysis. It was found that the total effort required to perform the analysis can be reduced by more efficiently leveraging the results of earlier analysis stages. For example, the specific types of process model flaws that are relevant to the defined hazards was found to be identifiable directly from the set of hazardous control actions by decomposing the actions using the general structure defined in chapter 3. To further reduce repetition in more detailed analyses, an approach was developed to perform the causal factor analysis based on specific types of process model flaws. Each specific process model flaw may be relevant to several different hazardous control actions, which might otherwise be analyzed separately resulting in the same causal factor being identified or re-identified many times throughout the analysis. The proposed procedures were found to reduce these kinds of repetitions in the analysis.

While chapter 3 proposed comprehensive procedures that provide more guidance to engineers and analysts performing a hazard analysis manually, chapter 4 formalized STPA and STAMP using logical and mathematical structures that can be used by automated tools. Several opportunities for automated methods to assist in performing the hazard analysis were discussed and a number of algorithms and formal methods were proposed. For example, given certain information, a set of potential hazardous control actions can be automatically generated for engineering analysis and review. The specific process model flaws that could be hazardous can be extracted automatically from the set of hazardous control actions, and complete traceability back to the system-level hazards can be generated. Some primitive basic scenarios can also be generated to guide the analysis; however human engineering analysis and review is required to identify the causal factors in most cases.

Chapter 4 also explored the potential for completely automatic hazard analysis and several major limitations were identified, particularly for early phases of development and systems in which no complete model of the system and its environment exists at the time of analysis. Techniques for partial automation to assist manual (human) analysis were developed and found to be appropriate for the systems studied so far.

A significant limitation to hazard analysis—the growing complexity of modern systems—was discussed in chapter 5 and several techniques to help manage and control complexity in the analysis were proposed. More efficient methods for performing subsequent low-level analysis iterations were developed, including methods that decompose the necessary tasks into manual and automated components for in-depth analyses. Several tools were also proposed to assist and interact with human engineers performing the analysis, some of which are currently in development. Finally, the proposed extensions were applied to more complex versions of the train door controller example used throughout this dissertation as well as the case study of the nuclear reactor control system.

There are many potential avenues for future work that builds on these extensions. Although more detailed procedures were introduced for identifying accident scenarios and causal factors, there is a potential for improvements that can further reduce repetition in the analysis and provide visualizations that better facilitate documentation and review of hazard analysis results. The work in [136] represents a promising start in this direction. In addition, there are a number of formal requirements analysis techniques in the literature (e.g. [143] and [148]); given the formal definitions in chapter 4, these existing techniques could be adapted or integrated into STPA to better study the necessary safety requirements and support low-level hazard analyses.

The preliminary tools defined in chapter 5 can also be improved, especially in terms of their interface and the organization of the results produced during an STPA analysis. Although chapters 3 and 4 address hazardous behavior and requirements related to timing information, the automated tools proposed in chapter 5 still need to incorporate this information. There is also the potential to develop new tools in addition to those in chapter 5, including tools that better facilitate the application of STPA Step 2 and potentially generate better visualizations like those in [136].

In conclusion, this dissertation opened the door to a number of new possibilities both in practice and in theory. For practitioners, chapter 3 provides more detailed guidance for those learning STPA hazard analysis and adds more rigor for those already engaged in applying STPA. For academics and theorists advancing state-of-the-art methods, chapter 4 describes a first step in formalizing STPA and defining automated methods that can detect requirements flaws and design conflicts from STPA results—even before detailed or executable models of the system exist. Chapter 5 attempts to leverage components from both worlds to propose useful and practical tools that can be implemented to improve safety today. Several organizations and industries from automotive to aviation have already begun applying these approaches, and new ideas and more powerful improvements are sure to follow. Whatever the future holds, it is my hope that this work will help stimulate new ideas in our community and that it has inspired you, the diligent reader, to go out and make a safer world!

[Page intentionally left blank]

Appendix A: Case Study of Nuclear Power Plant Systems

This appendix presents an analysis of a generalized version of an EPR (Evolutionary Power Reactor), a version of which appears in [140]. The EPR studied is a type of PWR (Pressurized Water Reactor). The system includes one Steam Generator (SG) and one Main Steam Isolation Valve (MSIV). The EPR reactor is fully digital, that is, all control systems, including the Reactor Protection System, are digital. The analysis focuses on a sub-set of the Nuclear Power Plant (NPP) system: the systems involved in closing the Main Steam Isolation Valve (MSIV). The same process could be applied to the rest of the system.

A generic diagram of a PWR is shown in Figure 38. During normal operation, the coolant in the primary cooling system (left of the diagram) transfers heat from the reactor to the Steam Generator (SG). The SG contains water that cools the primary coolant and evaporates into steam. The SG prevents primary coolant, which is radioactive, from mixing with the water, which is not radioactive. The steam produced in the SG travels to a turbine connected to a generator to produce electricity. The steam is cooled in the condenser and pumped back into the SG to begin the cycle again. The loop formed by the SG, turbine, and condenser is known as the secondary cooling system.

The MSIV is a valve located on the main steam line from the SG. During normal operation, the MSIV is kept open to permit cooling of the primary cooling system via the secondary system. In case of an abnormal situation, the MSIV can be closed to isolate the SG from the rest of the secondary system. MSIV closure is necessary if there is a break in the main feedwater pipe to the SG that allows water to leak out, an internal SG Tube Rupture (SGTR) that allows primary coolant to mix with secondary water, or a break in the main steam line exiting the SG.

Because MSIV closure prevents the secondary system from adequately cooling the primary system, a number of backup systems are provided to cool the primary coolant in case of MSIV closure. These backup systems include redundant SGs, turbine bypass valves, main steam relief isolation valves (MSRIV) and main steam relief control valves (MSRCV), safety relief valves (SRV), the Chemical Volume Control System (CVCS), and the Emergency Core Cooling System (ECCS). These systems are included in the analysis only to the extent that they impact the decision to close the MSIV.

The STPA analysis that follows begins by identifying the accidents, hazards, and control structure for the overall system. The remaining steps focus on those systems related to closure of the MSIV.

-

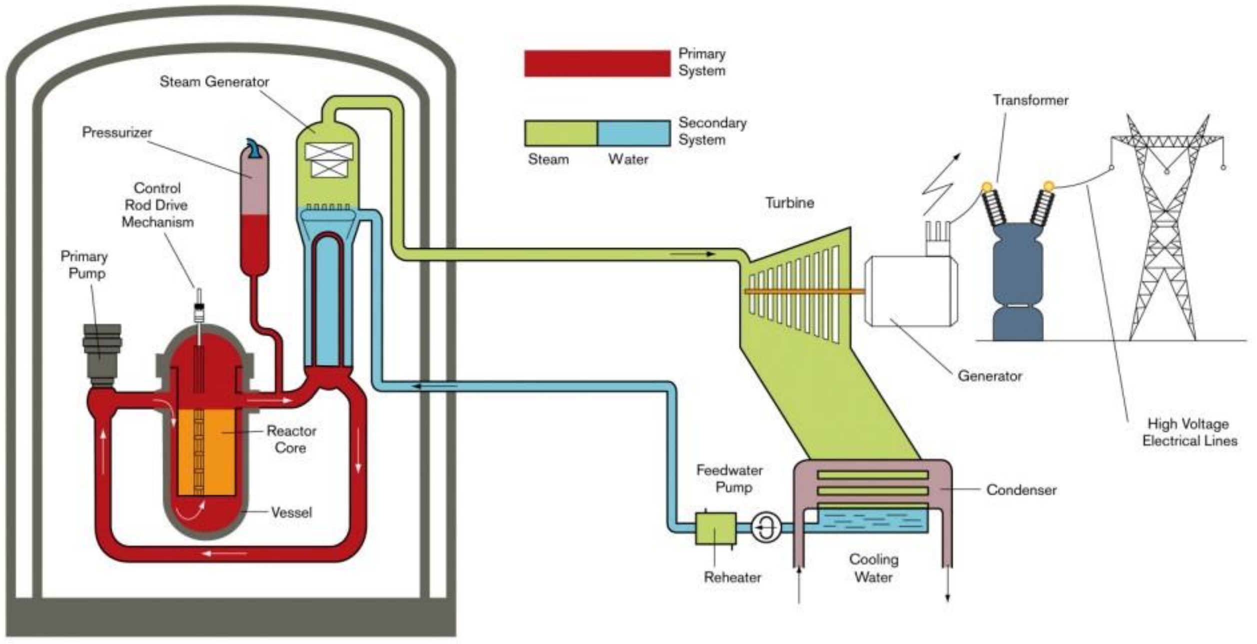


Figure 38: Pressurized Water Reactor (Diagram from [141])

A.1 Accidents

The first step is to identify the system-level losses, or accidents, to be considered. Accidents often involve loss of human life or injury, but any loss can be included that is unacceptable and must be prevented. Table 24 below shows the system-level accidents that are analyzed in this analysis.

Table 24: NPP system-level accidents to be prevented

A-1: People injured or killed
A-2: Environment contaminated
A-3: Equipment damage (economic loss)
A-4: Loss of electrical power generation

People injured or killed (A-1) includes both employees and the general population, and may involve radiation exposure, explosion, or any other mechanism. Environment contaminated (A-2) includes radiation or other harmful release to the air, ground, or groundwater, or any other part of the environment. Equipment damage (A-3) refers to the economic loss associated with any damage to equipment regardless of whether any radiation is released. Loss of electrical power generation (A-4) includes any unplanned plant shutdown.

Priorities may be assigned as not all accidents are equally important. In addition, the accidents are not mutually exclusive, and in fact it is possible to experience all four losses at once. Finally, economic damage such as equipment loss or the loss of electrical power generation (A-4) may not be of immediate importance in a licensing review or a traditional safety analysis but it is certainly a concern for the utility. STPA can be used for any type of loss that is important to those doing the analysis. Incorporating other types of losses, such as mission or economic losses, can not only allow better decision making with respect to achieving multiple requirements but can also assist in identifying and making tradeoffs between conflicting goals.

A.2 System Hazards

Once the system accidents have been defined, the hazards can be identified. Table 25 summarizes the hazards included in this analysis and the accidents to which they are related.

Table 25: NPP system-level hazards

Hazard	**Related Accident**
H-1: Release of radioactive materials	A-1, A-2
H-2: Reactor temperature too high	A-1, A-2, A-3, A-4
H-3: Equipment operated beyond limits	A-3, A-4
H-4: Reactor shut down	A-4

Release of radioactive materials (H-1) refers to any release outside the primary system, regardless of quantity, including releases into the secondary cooling system, groundwater, and air inside or outside the containment structure(s). These releases should be controlled to prevent exposure to people or the environment (A-1 and A-2). *Reactor temperature too high* (H-2) is a dangerous condition that can cause every system-level accident (for example, if the fuel rods melt), or it may lead to A-1 and A-2 without any radiation release (for example, through hydrogen production or other dangerous conditions).[29] Although H-2 may exist without an accident (for example, if there is a hydrogen explosion but containment holds), H-2 is a dangerous condition that should be controlled in the design. *Equipment operated beyond limits* (H-3) includes operation beyond safe limits that causes reactor damage or operation beyond design limits that causes damage to other equipment. *Reactor shut down* (H-4) includes any unplanned shutdown that may result in a loss of electrical power generation.

A.3 Safety Control Structure

The high-level safety control structure developed for this project is shown in Figure 39. The components inside the dashed (red) box control the closing of the MSIV. They are analyzed in further detail for the remainder of the case study. Figure 40 shows a more detailed control structure for the systems highlighted in the dashed box.

The dotted (green) arrow represents the communication between the MSIV controllers and other controllers. For example, the Protection System (PS) contacts the Safety Control System (SCS) in order to initiate the Engineering Safety Features (ESF) controls following ESF actuation. The Reactor Controls (RC) controller also communicates with Non-Safety System Controller (NSSC) in order to provide command signals for actuators used in RC functions other than control rods, such as the BMC (Boron and Makeup Control) components for Boron control.

There are four controllers that can provide a control action to close the MSIV: the Operator, the NSSC, the PS, and the Diverse Automation System (DAS). These four controllers send control actions to the MSIV Priority Module (PM), which uses a pre-programmed priority setting to determine which control actions to forward to the MSIV actuator. In this sense, the PM can also send control actions.

If the operator detects a need to close the MSIV, he or she may issue a *Close MSIV* command to the PM. The PM determines which controller is in charge according to a priority scheme, and forwards

[29] “Too high” is in relation to NRC standards and operation guidelines.

commands directly to the MSIV actuator. In this case, the PM would normally forward the command from the operator to the MSIV actuator.

The operator may also send a *Close MSIV* command to the NSSC, which provides manual control for the MSIV. In this situation, the NSSC would normally forward the command from the operator to the PM, which would then forward the command to the MSIV actuator.

The PS is an automated system that can automatically detect some situations in which a *Close MSIV* command is necessary. In these situations the PS can provide the *Close MSIV* command to the PM which can forward the command to the MSIV actuator.

Finally, the DAS is a backup protection system that is used if there is a problem with the PS. The DAS can issue a *Close MSIV* command to the PM, which would normally forward the command to the MSIV actuator.

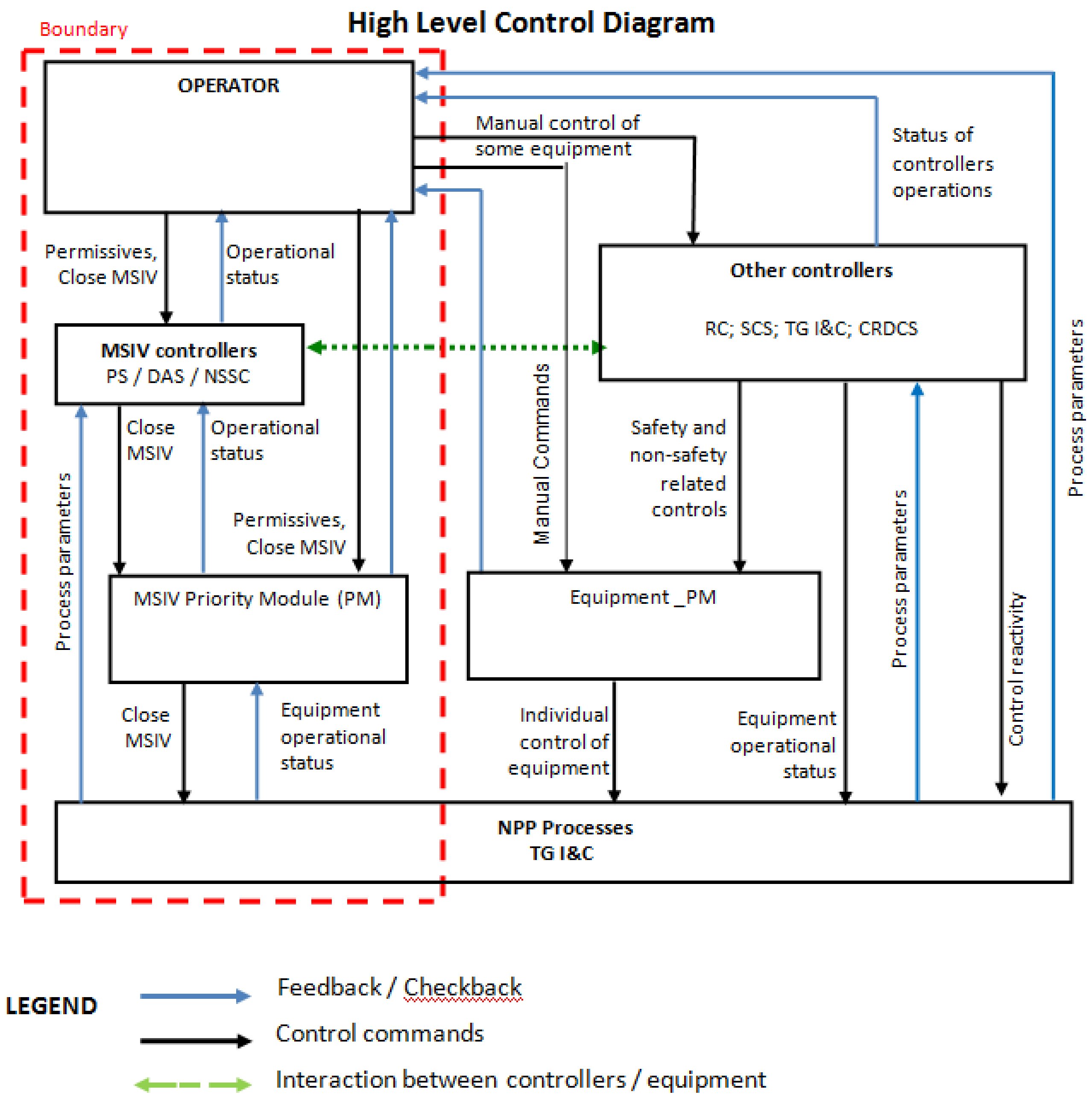


Figure 39: High-Level PWR Safety Control Structure

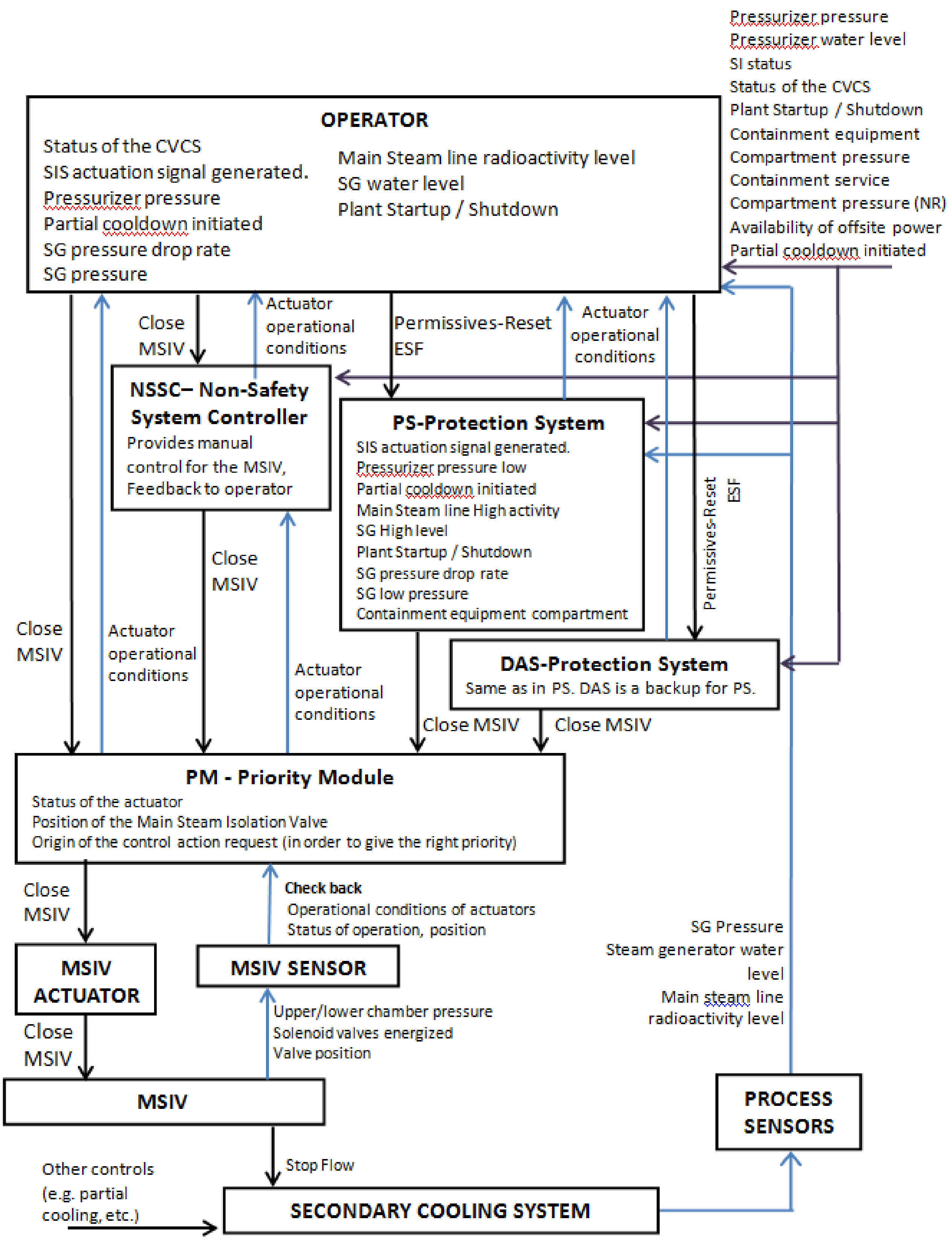

Pressurizer pressure
Pressurizer water level
SI status
Status of the CVCS
Plant Startup / Shutdown
Containment equipment
Compartment pressure
Containment service
Compartment pressure (NR)
Availability of offsite power
Partial cooldown initiated
OPERATOR
Status of the CVCS
SIS actuation signal generated.
Pressurizer pressure
Partial cooldown initiated
SG pressure drop rate
SG pressure
Main Steam line radioactivity level
SG water level
Plant Startup / Shutdown
Close MSIV
Actuator operational conditions
Permissives-Reset ESF
Actuator operational conditions
NSSC– Non-Safety System Controller
Provides manual control for the MSIV, Feedback to operator
PS-Protection System
SIS actuation signal generated.
Pressurizer pressure low
Partial cooldown initiated
Main Steam line High activity
SG High level
Plant Startup / Shutdown
SG pressure drop rate
SG low pressure
Containment equipment compartment
Permissives-Reset ESF
Close MSIV
Close MSIV
Actuator operational conditions
Actuator operational conditions
DAS-Protection System
Same as in PS. DAS is a backup for PS.
Close MSIV
Close MSIV
PM - Priority Module
Status of the actuator
Position of the Main Steam Isolation Valve
Origin of the control action request (in order to give the right priority)
Close MSIV
Check back
Operational conditions of actuators
Status of operation, position
MSIV ACTUATOR
MSIV SENSOR
SG Pressure
Steam generator water level
Main steam line radioactivity level
Close MSIV
Upper/lower chamber pressure
Solenoid valves energized
Valve position
MSIV
PROCESS SENSORS
Other controls (e.g. partial cooling, etc.)
Stop Flow
SECONDARY COOLING SYSTEM

Figure 40: More detailed safety control structure for MSIV control

A sensor provides feedback about the MSIV status directly to the PM. This sensor does not sense process variables such as pressure, temperature, or steam flux. Instead, it senses torque applied to the valve itself to detect if the valve has closed. The PM receives this feedback and can provide confirmation back to the controller that originally requested the MSIV closure.

Other process sensors report process variables to the controllers including various pressures, SG water level, and the operation of other backup systems. This information is used by the controllers to determine, among other things, whether the MSIV should be closed.

A.4 Controller Responsibilities

The controllers have responsibilities as follows:

Operator :

- Validate/inhibit permissives
- Bring the plant to a controlled shutdown in case of Anticipated Operational Occurrence (AOO) or Postulated Accidents (PA), such as leakage from primary into the secondary loop.
- Activate the safety engineering features (ESF)
- Start main steam line isolation when necessary
- Monitor parameters and look for abnormalities or trends (fault diagnostic)
- Operate the plant during startup
- Operate the plant during programmed shutdown
- Take actions in accordance to written guides upon any transient or emergency

PS - Protection System:

- Bring the plant to a controlled shutdown in case of Anticipated Operational Occurrence (AOO) or Postulated Accidents (PA), such as leakage from primary into the secondary loop.
- Activate the safety engineering features (ESF)
- Start main steam line isolation when necessary

DAS - Diverse Automation System

- Same as PS. DAS is a backup for PS.

NSSC - Non-Safety System Controller

- If an operator command to open/close MSIV is received, then send that command to PM
- If feedback is received from PM, then send that feedback to Operator.

PM - Priority Module

- Select one controller to be active
- Forward commands to MSIV actuator
- Forward feedback from MSIV actuator to the active controller
- Ensure that checkback is received when MSIV is closed (indicating that valve torque has reached its maximum)
- Check for any problems with MSIV actuator operability

A.5 Process Model Variables

The process model variables capture the information needed by each controller to decide what control action to provide. Different process model variables may be associated with each control action.

The high-level process model variables associated with MSIV closure can be identified by considering the purpose of the MSIV. The MSIV remains open during normal plant operation and is only needed to control a few specific abnormal conditions. The relevant high-level conditions can be derived from the system hazards and system description as follows:[30]

- Steam generator tube rupture, which can cause an uncontrolled SG level increase and can release contaminated fluid into the secondary system
- Steam system piping leak, which can depressurize the SG and cause an overcooling transient and energy release into containment
- Feedwater system piping leak, which can depressurize the SG and cause an overcooling transient and energy release into containment

[30] See also [142] chapter 7 pages 7.3-22 and 7.3-11

While these conditions could be caused by physical failures, the latter two could also be caused by design flaws or unsafe commands elsewhere in the system. For example, a leak in the main steam line could be caused by a physical failure (e.g. rupture in the line) or it could be caused by main steam relief valves that are opened inadvertently or at the wrong time. Both situations could require MSIV closure to prevent depressurization and an overcooling transient while the issue is investigated and resolved.

In addition to helping to mitigate the conditions above, the MSIV also controls the heat exchange that takes place within the SG. Before the SG is closed, other support systems[31] may need to be engaged to provide the additional cooling needed. Therefore, information about additional cooling provided by other support systems (i.e. inadequate, adequate[32]) may be needed for the decision to close the MSIV and should be included in the process model.

A.6 Unsafe Control Actions

When considering whether a potential control action is hazardous or not, it is important to avoid assuming that other defense barriers are intact or that they are appropriate, sufficient, and error-free. For example, even if there is an emergency feedwater system to provide the necessary cooling in the event of a relief valve inadvertently commanded open, it is still hazardous to inadvertently command the relief valve open. These hazardous actions must be included in the analysis and prevented regardless of other protective systems intended to mitigate unsafe behavior.

Table 26 summarizes the hazardous control actions that were identified for the command *Close MSIV*.

[31] *Other support systems* refers to other components designed to cool the primary system. These include the CVCS, SI, CCS, etc.

[32] *Adequate* means the system operation is sufficient to provide the cooling normally provided by the SG.

Table 26: Hazardous Control Actions for Close MSIV

Control Action	**Hazardous Control Actions**			
	Not Providing Causes Hazard	**Providing Causes Hazard**	**Wrong Timing or Order Causes Hazard**	**Stopped Too Soon or Applied Too Long**
Close MSIV	Close MSIV not provided when there is a rupture in the SG tube, leak in main feedwater, or leak in main steam line [H-2, H-1, H-3]	Close MSIV provided when there is no rupture or leak [H-4] Close MSIV provided when there is a rupture or leak while other support systems are inadequate [H-1, H-2, H-3]	Close MSIV provided too early (while SG pressure is high): SG pressure may rise, trigger relief valve, abrupt steam expansion [H-2, H-3] Close MSIV provided too late after SGTR: contaminated coolant released into secondary loop, loss of primary coolant through secondary system [H-1, H-2, H-3] Close MSIV provided too late after main feedwater or main steam line leak [H-1, H-2, H-3, H-4]	N/A

The hazardous control actions in Table 26 were identified using the following process. First, a controller and control action were selected. The operator and the control action *Close MSIV* were analyzed first, although the results also apply to other controllers in the system. A context table was then constructed for the control action using the corresponding process model variables that were defined previously. Table 27 shows the context table for *Close MSIV provided*.

Table 27: Context table for *Operator provides Close MSIV* control action

	1	2	3	4	5	6	7	8
	Control Action	**Steam Generator Tube**	**Condition of Main Feedwater Pipe**	**Condition of Main Steamline**	**Operation of other support systems**	**Control Action Hazardous?**	**Control Action Hazardous if Too Late?**	**Control Action Hazardous if Too Early?**
1	*Close MSIV*	Not Ruptured	No Leak	No Leak	Adequate	H-4	H-4	H-4
2		Ruptured	No Leak	No Leak	Adequate	No	H-1, H-2, H-3, H-4	H-3, H-4
3		Not Ruptured	Leak	No Leak	Adequate	No	H-2, H-3, H-4	No
4		Not Ruptured	No Leak	Leak	Adequate	No	H-2, H-3, H-4	No
5		Ruptured	Leak	No Leak	Adequate	No	H-1, H-2, H-3, H-4	H-3, H-4
6		Not Ruptured	Leak	Leak	Adequate	No	H-2, H-3, H-4	No
7		Ruptured	No Leak	Leak	Adequate	No	H-1, H-2, H-3, H-4	H-3, H-4
8		Ruptured	Leak	Leak	Adequate	No	H-1, H-2, H-3, H-4	H-3, H-4
9		Not Ruptured	No Leak	No Leak	Inadequate	H-2, H-4	H-2, H-4	H-2, H-4
10		Ruptured	No Leak	No Leak	Inadequate	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4
11		Not Ruptured	Leak	No Leak	Inadequate	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4
12		Not Ruptured	No Leak	Leak	Inadequate	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4
13		Ruptured	Leak	No Leak	Inadequate	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4
14		Not Ruptured	Leak	Leak	Inadequate	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4
15		Ruptured	No Leak	Leak	Inadequate	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4
16		Ruptured	Leak	Leak	Inadequate	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4	H-1, H-2, H-3, H-4

Column 1 in Table 27 is the control action being analyzed while columns 2 to 5 correspond to the process model variables identified in section A.5. Column 6 specifies in which contexts it is hazardous to provide the *Close MSIV* control action. For example, row 1 describes a situation in which it is hazardous to close the MSIV: if there is no SG tube rupture, no main feedwater pipe leak, and no main steam line leak, then there is no need to close the MSIV. Closing the MSIV will cause H-4 (reactor shut down). If the operation of other support systems cannot make up for the additional heat exchange required, closing the MSIV will also lead to a loss of necessary cooling (H-2 in row 9 column 6).

If other support systems, including other CVCS, SI, ECCS, etc., are producing the additional cooling required during a rupture/leak, then closing the MSIV is not hazardous (rows 2-8, column 6) and a reactor shutdown is initiated regardless of any MSIV actions. If for some reason the other systems are not capable of producing the additional cooling needed, then closing the MSIV may cause other hazards (rows 10-16, column 6) including excessive temperature increase (H-2), release of radioactive materials (H-1), an immediate reactor shutdown or SCRAM (H-4) if not already triggered, and additional equipment damage (H-3). Depending on the type of rupture, it may actually be better to keep the MSIV open to control the temperature of the reactor (H-2) even though that would permit some radioactive steam to be introduced into the secondary system (H-1).

The last two columns on the right in Table 27 take into account timing information. If there is a rupture/leak and other support systems are adequate, then it is not hazardous to close the MSIV (e.g. row 2-8). The MSIV should be closed. However, if the MSIV is closed *too late* in this context then it is hazardous. If the steam generator tube is ruptured, too much radioactive coolant may have already been released into the secondary system and the environment (H-1). If the steam line has a leak, excessive steam may have been released causing overcooling and overcompensation (H-2). If the steam line or feedwater pipe have a leak, the SG may run dry and cause equipment damage (H-3). Closing the MSIV *too early* may also be hazardous in some situations. For example, if the steam generator tube is ruptured then the SG pressure should be decreased before the MSIV is closed. Otherwise, if the MSIV is closed too early after a SG tube rupture, then the SG pressure and temperature will increase and may cause equipment damage to the SG, SG piping, or other systems (H-3).

The contexts used to define hazardous control actions may not be the same as contexts that are inherently unsafe. The tables in this section are used to analyze controller behavior and control actions in a number of contexts, not to analyze contexts that are unsafe by themselves. For example, row 1 column 6 of Table 27 is marked as hazardous because the control action *Close MSIV* will cause a hazard if provided in that context, even though the context by itself (no ruptures/leaks) does not describe anything hazardous. Conversely, the context in row 2 describes a steam generator tube rupture but column 6 is not marked as hazardous because closing the MSIV is not a hazardous behavior in that context. In fact, closing the MSIV is exactly what should happen in that situation to prevent an accident.

Although providing a control action can be hazardous, *not* providing a control action can be equally hazardous. Table 28 shows the context table for *not* providing the *Close MSIV* control action. As before, a reactor shutdown should be initiated for any rupture regardless of the MSIV control action. However because these tables are used to identify hazardous control actions, only hazards that are affected by an absent *Close MSIV* control action are listed at this stage of the analysis.

If there is no rupture/leak, keeping the MSIV open is not hazardous (rows 1 and 9). However, if there is a rupture/leak, different hazards may be experienced depending on what part of the system is affected. If the SG tube is ruptured and the MSIV is not closed, radioactive material will be released into the secondary system (H-1) and the SG water level may increase uncontrollably. A sustained release of primary coolant will decrease the effectiveness of the primary cooling system (H-2), and the release of radioactive material into the secondary system may cause equipment damage (H-3). If the main steam line has a leak and the MSIV is not closed, excessive steam may be released causing an overcooling transient and overcompensation by other systems to increase reactivity (H-2). Excessive steam release may also lower the SG water level, causing potential equipment damage if the SG runs dry (H-3). If the main feedwater pipe has a leak and the MSIV is not closed, the SG may be depressurized causing an overcooling transient and water level may drop, leading to H-2 and H-3 as above.

Table 28: Context table for *Operator does not provide Close MSIV control action*

	1	2	3	4	5	6
	Control Action	**Steam Generator Tube**	**Condition of Main Feedwater Pipe**	**Condition of Main Steamline**	**Operation of other support systems**[33]	**Not Providing Control Action is Hazardous?**
1	*Close MSIV*	Not Ruptured	No Leak	No Leak	Adequate	No
2		Ruptured	No Leak	No Leak	Adequate	H-1, H-2, H-3, H-4
3		Not Ruptured	Leak	No Leak	Adequate	H-2, H-3
4		Not Ruptured	No Leak	Leak	Adequate	H-2, H-3
5		Ruptured	Leak	No Leak	Adequate	H-1, H-2, H-3, H-4
6		Not Ruptured	Leak	Leak	Adequate	H-2, H-3
7		Ruptured	No Leak	Leak	Adequate	H-1, H-2, H-3, H-4
8		Ruptured	Leak	Leak	Adequate	H-1, H-2, H-3, H-4
9		Not Ruptured	No Leak	No Leak	Adequate	No
10		Ruptured	No Leak	No Leak	Inadequate	H-1, H-2, H-3, H-4
11		Not Ruptured	Leak	No Leak	Inadequate	H-2, H-3
12		Not Ruptured	No Leak	Leak	Inadequate	H-2, H-3
13		Ruptured	Leak	No Leak	Inadequate	H-1, H-2, H-3, H-4
14		Not Ruptured	Leak	Leak	Inadequate	H-2, H-3
15		Ruptured	No Leak	Leak	Inadequate	H-1, H-2, H-3, H-4
16		Ruptured	Leak	Leak	Inadequate	H-1, H-2, H-3, H-4

[33] *Other support systems* refers to other systems designed to cool the primary system. This includes the CVCS, SI, CCS, etc. *Adequate* means the system operation is sufficient to provide the cooling normally provided by the SG.

In the case of SG tube rupture, keeping the MSIV open can cause not only equipment damage but also a more immediate shutdown (H-4) via SCRAM and can increase the amount of time the plant will need to remain shut down for repairs. The overfilling of the SG could allow water to enter the steam lines, damaging the delicate turbine pallets and requiring extensive time for repairs. In addition to actual damage, equipment can be overstressed and require more detailed inspections before the plant can be operational again. The additional contamination will also require more time to decontaminate and will result in the generation of more waste. Because keeping the MSIV open during a SG tube rupture will cause a more severe and prolonged shutdown than would otherwise occur with a contained SG tube rupture, H-4 is included in Table 28 for these cases. H-4 is not listed for other cases because it is assumed that keeping the MSIV open after a leak in the main steamline or main feedwater pipe will not cause a more severe or prolonged shutdown than if the MSIV is closed, although it does contribute to the other hazards listed.

Note that for the purpose of reviewing the tables, the rationale behind each of the “hazardous” vs. “not hazardous” decisions should be documented during the analysis. In fact, the context tables can be used to help verify that the necessary rationales and assumptions are documented during the analysis, as opposed to ad-hoc identification of hazardous control actions that may immediately discount and omit non-hazardous control actions entirely. Of course, the non-hazardous rows could easily be omitted from the context tables if desired; however, documenting the conclusions about what behavior is hazardous can be just as important as documenting behavior that is assumed to be non-hazardous. Such documentation may be especially important for other long-term project goals like future change management activities, design re-use in new environments, and other considerations that arise later in the system lifecycle.

A comparison of Table 11 and Table 12 shows that there are conflicts that must be resolved. In both tables, rows 10 to 16 are marked as hazardous. In other words, in these situations it is hazardous to close the MSIV yet hazardous to keep the MSIV open. In some cases, it is possible to revisit the design to eliminate the conflict and provide a safe option. If the conflict cannot be resolved, a decision must be made about what action should be taken in these contexts, that is, which is the *least* hazardous? For this case study, after consultation with nuclear engineers and regulators it was found that rows 10 to 16 may not have been analyzed in previous safety analyses with respect to MSIV control. For the purposes of this research, the consensus was to assume that it may be best to keep the MSIV open in the context of row 10 to maximize the amount of cooling provided even though doing so will contaminate the secondary cooling system and eventually require costly repairs. Rows 11-16, on the other hand, involve leaks in the pipe supplying water to the steam generator and/or the line that carries steam away. If the MSIV is left open in these situations, the amount of water in the steam generator can decrease and eventually lead to less cooling capability or an overcooling transient. Therefore, in these situations (rows 11-16), it was assumed that it may be best to keep the MSIV closed to maximize the amount of cooling provided even though it is only a temporary measure. These solutions were found to differ from current designs of MSIV controllers, which do not act based on the state of other support systems and may automatically close the MSIV during any rupture. Chapter 4 discusses design conflicts in more detail, including search and detection methods that can be performed on STPA results to automatically detect such conflicts.

Both of these assumptions should be reviewed and evaluated carefully by domain experts. The purpose of this research case study was not to provide final solutions to these hazardous situations, but to develop and apply hazard analysis methods that can uncover hazardous control and provide the safety-critical questions that need to be considered. Note that although Table 11 and Table 12 use high-level contexts, the analysis can also be performed in more detail using the techniques described in chapter 5. A more detailed analysis could be necessary if, for example, it is found that the best solution depends on the type of steam generator tube rupture, the amount of pressure in the SG, etc.

Of course, in any of these situations, there are other control actions that need to take place outside the MSIV control loop—they can be analyzed using the same approach. In addition, every effort should be made to prevent many of these contextual conditions from existing in the first place. Although such additional efforts were outside the scope of this initial case study, they are mentioned here to show how the analysis may branch out into other areas of the system to address the issues identified.

A.7 Safety Constraints

Once conflicts are resolved as discussed in the previous section, the remaining hazardous control actions can be summarized and translated into safety constraints as shown in Table 29.

Table 29: Post-conflict-resolution unsafe control actions and safety constraints

Unsafe Control Action	**Safety Constraint**
UCA 1: Close MSIV not provided when there is a leak (rupture in the SG tube, leak in main feedwater, or leak in main steam line) and the support systems are adequate	**SC 1**: MSIV must be closed when there is a leak (rupture in the SG tube, leak in main feedwater, or leak in main steam line) and the support systems are adequate
UCA 2: Close MSIV not provided when there is a main feedwater or main steam line leak and other support systems are inadequate	**SC 2**: MSIV must be closed when there is a main feedwater or main steam line leak and other support systems are inadequate
UCA 3: Close MSIV provided when there is a SGTR but support systems are inadequate	**SC 3**: MSIV must not be closed when there is a SGTR and support systems are inadequate
UCA 4: Close MSIV provided too early (while SG pressure is high)	**SC 4**: MSIV must not be closed too early while SG pressure is too high
UCA 5: Close MSIV provided too late after rupture/leak (in the SG tube, main feedwater, or main steam line)	**SC 5**: MSIV must not be closed too late after rupture/leak (in the SG tube, main feedwater, or main steam line)
UCA 6: Close MSIV provided when there is no rupture/leak	**SC 6**: MSIV must not be closed when there is no rupture/leak

A.8 Causal Factors

As described in earlier, there are two ways that a safety constraint can be violated:

1. The controller provides an unsafe control action
2. Appropriate control actions are provided but not followed

The causal factors shown in Figure 41 are used for the analysis in this case study. The following sections analyze both cases for the Operator, DAS, and PS.

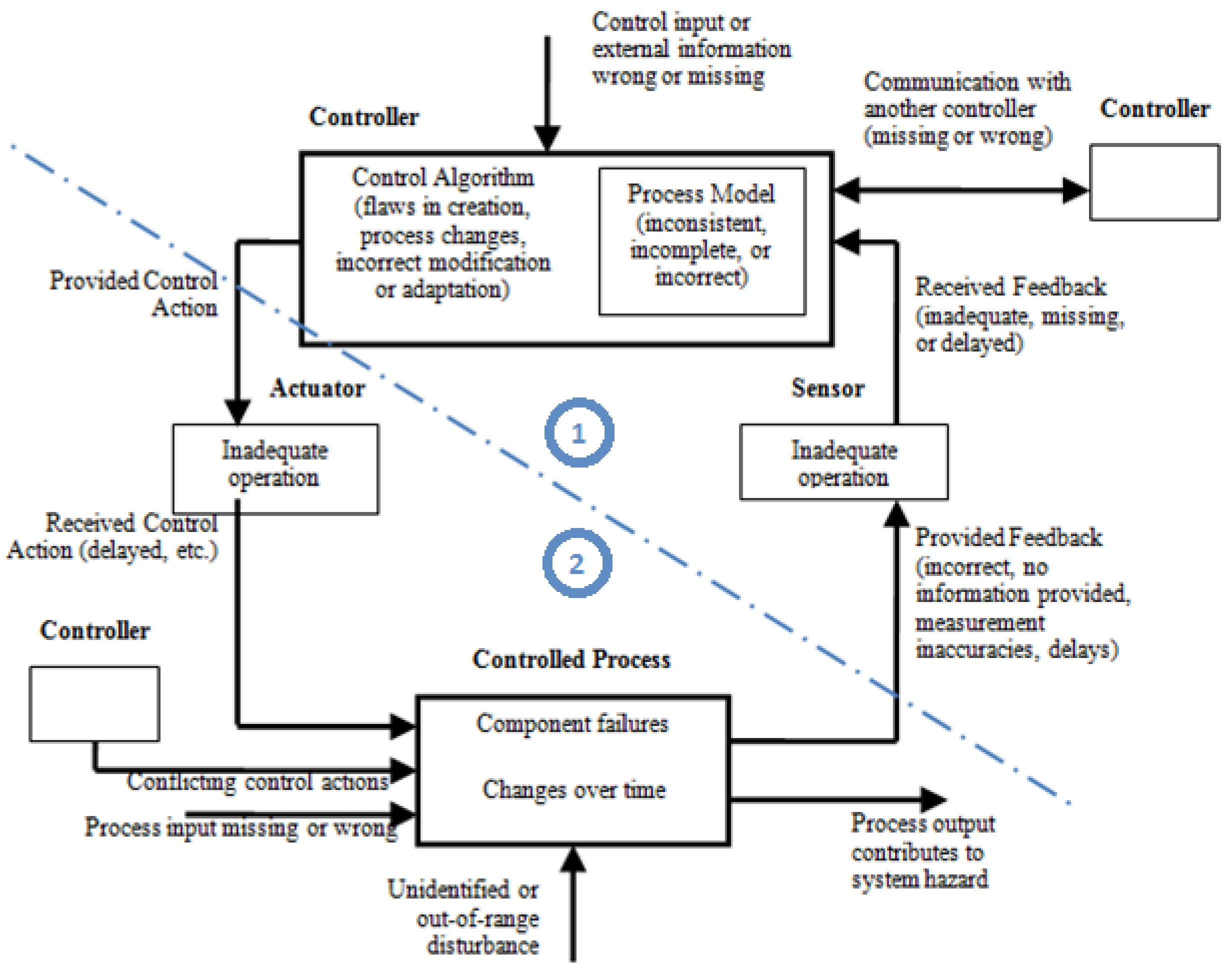
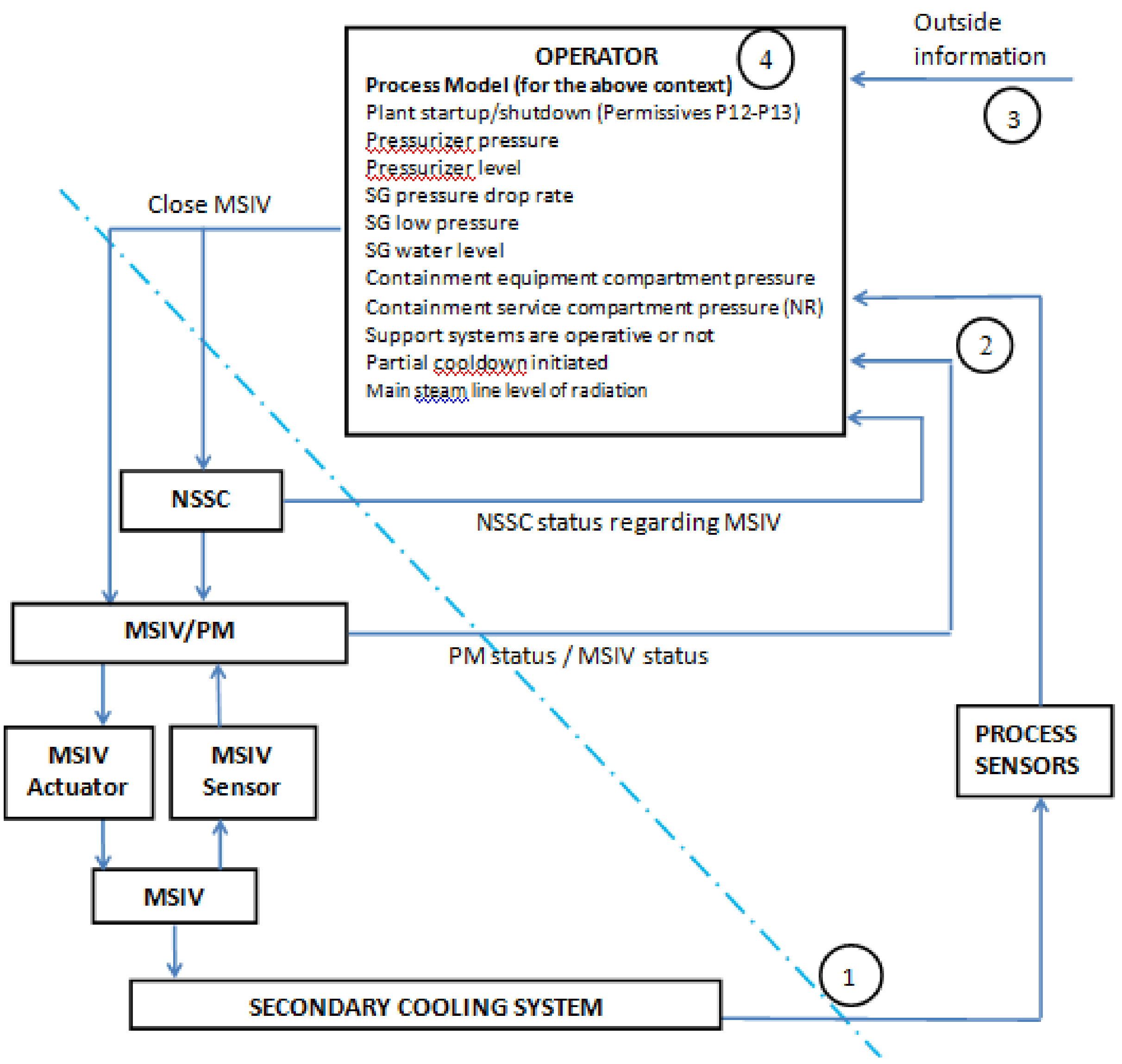

Control input or external information wrong or missing
Controller
Communication with another controller (missing or wrong)
Controller
Control Algorithm (flaws in creation, process changes, incorrect modification or adaptation)
Process Model (inconsistent, incomplete, or incorrect)
Provided Control Action
Received Feedback (inadequate, missing, or delayed)
Actuator
Sensor
1
Inadequate operation
Inadequate operation
Received Control Action (delayed, etc.)
2
Provided Feedback (incorrect, no information provided, measurement inaccuracies, delays)
Controller
Controlled Process
Component failures
Changes over time
Conflicting control actions
Process input missing or wrong
Process output contributes to system hazard
Unidentified or out-of-range disturbance

Figure 41: A classification of causal factors leading to hazards

A.8.1 Operator Causal Factors

A.8.1.1 Causal Factors Leading to Operator Unsafe Control Actions

This section identifies causal factors that can lead to each unsafe control action summarized in Figure 29 for the Operator.



OPERATOR
4
Process Model (for the above context)
Plant startup/shutdown (Permissives P12-P13)
Pressurizer pressure
Pressurizer level
SG pressure drop rate
SG low pressure
SG water level
Containment equipment compartment pressure
Containment service compartment pressure (NR)
Support systems are operative or not
Partial cooldown initiated
Main steam line level of radiation
Outside information
3
Close MSIV
2
NSSC
NSSC status regarding MSIV
MSIV/PM
PM status / MSIV status
MSIV Actuator
MSIV Sensor
MSIV
PROCESS SENSORS
1
SECONDARY COOLING SYSTEM

Figure 42: Causal factors leading to operator unsafe control actions

<u>UCA 1</u>: Close MSIV command not provided when there is a leak (rupture in the SG tube, link in main feedwater, or leak in main steam line) and the support systems are adequate.

(1) Secondary cooling system (CVCS or emergency feedwater system)

   a. Concurrent situation masks another. For example, a feedwater problem could happen concurrent with a SGTR, causing the SG water level to stay practically stable.
   b. Situation that requires MSIV closure is masked. For example, NSSC engages PZR heaters to make up for loss of RCS pressure during SGTR.
   c. Event progresses too slowly to detect

(2) Process Feedback

   a. SG level feedback missing, delayed, or incorrect
   b. SG Pressure, or setpoints, is not correct or delayed
   c. Steam generator water level delayed or incorrect
   d. Main steam line activity not correctly indicated
   e. Conflicting data indicating a false situation
   f. Voting system does not operate properly and gives wrong measures
   g. No indication of partial cool down initiated
   h. Failures in sensors, communication lines, or power
   i. PM reports both MSIV actuators as inoperable when they are
   j. PM reports MSIV already closed, when it is not
   k. NSSC reported as operational (or no feedback provided) when it is not

(3) Outside information

   a. PZR pressure delayed or missing
   b. PZR level incorrectly indicated as normal
   c. No indication of SI initiated
   d. Delayed indication of SI initiated

e. Inappropriate permissives in effect[34]
f. Wrong combination of indicators from the 4 divisions

(4) Operator

a. Operator believes Steam Generator is not ruptured when it is ruptured
b. Operator believes the main steam line has no leak when it has a leak
c. Operator believes the main feedwater has no leak when it has a leak
d. Operator confused about the procedure to be followed
e. Operator confused because of conflicting indicators[5]
f. Operator reluctant to shutdown the reactor, unsure if shutdown is necessary or warranted
g. Operator under pressure not to trip reactor
h. Operator waits for the PS to handle the situation (e.g. Operator recognizes possible SGTR but believes PS will handle it)
i. Operator is not aware of the problem due to inadequate feedback (e.g. screen is frozen)
j. Operator is not aware because NSSC is inoperative or providing inadequate information
k. Operator closes the wrong valve
l. Operator recognizes the rupture/leak but believes other support systems are inadequate, and keeps MSIV open to maintain sufficient cooling capability.
m. Operator uncertain whether a rupture/leak exists (there is a conflict between being conservative under uncertainty versus immediate manual spurious shutdown which costs money and may be discouraged. May also prefer to wait for the automated system to resolve the problem versus intervening under uncertainty)

[34] One of the causes for wrong command can be confusion about indicators. “Confusion" could mean the model is not clear, there is an overlap of responsibilities, or conflicting process values are indicated. The controllers check several indicators to decide what the specific problem is. For example, the main steam pipe break would also cause high pressure in the main steam line compartment and low SG water level. However, SG low level together with permissive 13 (startup) may indicate there is no need to isolate the SG. It could happen that there is a problem with the sensors, the model (inside the controller) could be wrong, or the algorithm could be wrong

n. Operator believes NSSC is operational when it is not (could cause operator to provide command to an inoperative or disabled NSSC instead of directly to PM)

UCA 2: Close MSIV command not provided when there is a main feedwater or main steam line leak and other support systems are inadequate.

(1) Secondary cooling system (CVCS or emergency feedwater system)

a. Concurrent situation masks another. For example, a feedwater problem could happen concurrent with a SGTR, causing the SG water level to stay practically stable.

b. Situation that requires MSIV closure is masked.

c. Event progresses too slowly to detect

(2) Process Feedback

a. SG level feedback missing, delayed, or incorrect

b. SG Pressure, or setpoints, is not correct or delayed

c. Steam generator water level delayed or incorrect

d. Conflicting data indicating a false situation

e. Voting system does not operate properly and gives wrong measures

f. No indication of partial cool down initiated

g. Failures in sensors, communication lines, or power

h. PM reports both MSIV actuators as inoperable when they are

i. PM reports MSIV already closed, when it is not

j. NSSC reported as operational (or no feedback provided) when it is not

(3) Outside information

a. PZR pressure delayed or missing

b. PZR level incorrectly indicated as normal

c. No indication of SI initiated

d. Delayed indication of SI initiated

e. Inappropriate permissives in effect[35]
f. Wrong combination of indicators from the 4 divisions

(4) Operator

a. Operator believes the main steam line has no leak when it has a leak
b. Operator believes the main feedwater has no leak when it has a leak
c. Operator believes there is an SGTR that does not require MSIV closure when there is actually a main steam line or main feedwater leak that does require MSIV closure
d. Operator confused about the procedure to be followed
e. Operator confused because of conflicting indicators[5]
f. Operator reluctant to shutdown the reactor, unsure if shutdown is necessary or warranted
g. Operator under pressure not to trip reactor
h. Operator waits for the PS to handle the situation (e.g. Operator recognizes possible leak but believes PS will handle it)
i. Operator is not aware of the problem due to inadequate feedback (e.g. screen is frozen)
j. Operator is not aware because NSSC is inoperative or providing inadequate information
k. Operator closes the wrong valve
l. Operator recognizes the rupture/leak but because other support systems are inadequate, keeps MSIV open in an effort to maintain sufficient cooling capability.

[35] One of the causes for wrong command can be confusion about indicators. “Confusion" could mean the model is not clear, there is an overlap of responsibilities, or conflicting process values are indicated. The controllers check several indicators to decide what the specific problem is. For example, the main steam pipe break would also cause high pressure in the main steam line compartment and low SG water level. However, SG low level together with permissive 13 (startup) may indicate there is no need to isolate the SG. It could happen that there is a problem with the sensors, the model (inside the controller) could be wrong, or the algorithm could be wrong

m. Operator uncertain whether a rupture/leak exists (there is a conflict between being conservative under uncertainty versus immediate manual spurious shutdown which costs money and may be discouraged. May also prefer to wait for the automated system to resolve the problem versus intervening under uncertainty)
n. Operator believes NSSC is operational when it is not (could cause operator to provide command to an inoperative or disabled NSSC instead of directly to PM)

<u>**UCA 3**</u>: Close MSIV provided when there is SGTR but other support systems are inadequate

(1) Secondary cooling system
   a. A concurrent situation could mask another, other support systems could appear adequate but may not be, and automated systems could exacerbate the situation. For example, main steam line high radioactivity may be detected coincident with safety injection, making it difficult to detect whether partial cooldown was initiated by the automation.
   b. Loss of power

(2) Process Feedback
   a. SG level feedback not provided, delayed, or incorrect
   b. SG Pressure or setpoints are not correct, delayed, or missing
   c. Steam generator water level not correct, delayed, or missing
   d. Conflicting data indicating a false situation
   e. Voting system does not operate properly and gives wrong measures
   f. Failures in sensors, communication lines, or power

(3) Outside information
   a. Wrong combination of indicators from the 4 divisions
   b. PZR pressure delayed or missing
   c. False signal SI initiated

(4) Operator
   a. Operator thinks support systems are working when they are not. For example, NSSC may appear to be working but may not be because the screen is frozen. The

operator may believe that a partial cool down was initiated by the automation because safety injection was engaged at the same time that main steam line radioactivity was detected

b. Operator believes there is a main steam line or feedwater leak when there is actually an SGTR
c. Operator knows support systems are working, but does not realize they are inadequate
d. Operator confused about the procedure to be followed
e. Operator confused because of conflicting indicators
f. Operator does not realize other support systems are not operative (e.g. for maintenance or other reasons)

<u>UCA 4</u>: Close MSIV provided too early (while SG pressure is high)

(1) Secondary cooling system
  a. A concurrent situation could mask another. For example, a feedwater problem could happen concurrently with a SGTR, and the SG water level stay practically stable.
  b. Event progress too slowly to detect
  c. Actuation of NSSC could confuse Operator. For example, PZR heaters could make up for loss of RCS pressure

(2) Process Feedback
  a. SG level feedback not provided
  b. SG Pressure, or setpoints, is not correct
  c. Steam generator water level not correctly indicated
  d. Main steam line activity not correctly indicated
  e. Conflicting data indicating a false situation
  f. Voting system does not work properly and gives wrong measures
  g. Sensors failure

(3) Outside Information

a. PZR pressure delayed
b. PZR feedback missing
c. False feedback indicates PZR level is normal
d. No indication of SI initiated
e. No indication of partial cool down initiated
f. Permissives wrongly in effect[36]
g. Wrong combination of indicators from the 4 divisions

(4) Operator

a. Operator believes it is already safe to initiate action after indications confirm SGTR
b. Operator believes it is already safe to initiate action after indications confirm Main steam line break
c. Operator believes it is already safe to initiate action after indications confirm main feedwater break
d. Operator confused about the procedure to be followed
e. Operator confused because of conflicting indicators

UCA 5: Close MSIV command provided too late after rupture/leak (in the SG tube, main feedwater, or main steam line)

(1) Secondary cooling system

[36] One of the causes for wrong command can be confusion about indicators. "Confusion" could mean the model is not clear, there is an overlap of responsibilities, or conflicting process values are indicated. The controllers check several indicators to decide what the specific problem is. For example, the main steam pipe break would also cause high pressure in the main steam line compartment and low SG water level. However, SG low level together with permissive 13 (startup) may indicate there is no need to isolate the SG. It could happen that there is a problem with the sensors, the model (inside the controller) could be wrong, or the algorithm could be wrong

a. A concurrent situation could mask another one. For example, a feedwater problem could happen concurrently with a SGTR such that the SG water level stays practically stable.
b. Event progress too slowly to detect
c. Actuation of NSSC could confuse Operator. For example, PZR heaters could make up for loss of RCS pressure

(2) Process Feedback

a. SG level feedback not provided
b. SG Pressure, or setpoints, is not correct
c. Steam generator water level delayed
d. Main steam line activity not correctly indicated or delayed
e. Conflicting data indicating a false situation
f. Voting system does not work properly and gives wrong measures
g. Sensor failure
h. PM reports both MSIV actuators as inoperable when they are
i. PM reports MSIV as already closed, when it is not
j. NSSC reported as operational (or no feedback) when it is not

(3) Outside Information

a. PZR pressure delayed
b. PZR feedback missing
c. False feedback indicates PZR level is normal
d. No indication or delayed indication of SI initiated
e. No indication or delayed indication of partial cool down initiated
f. Permissives wrongly in effect
g. Wrong combination of indicators from the 4 divisions
h. Screen is blank or frozen/NSSC or PS provides no feedback

(4) Operator

a. Operator thinks it is not yet safe to initiate action after SGTR is confirmed
b. Operator thinks it is not yet safe to initiate action after main steam line leak is confirmed

c. Operator thinks it is not yet safe to initiate action after main feedwater leak is confirmed
d. Operator confused about the procedure to be followed
e. Operator confused because of conflicting indicators
f. Operator reluctant whether to shutdown the reactor
g. Operator under pressure not to trip reactor
h. Operator has a conflict between being conservative with uncertainty of whether there is a SGTR, or to do what it is expected, i.e. to wait for the automated system to resolve the problem. In other words, the operator tries to avoid spurious shutdown, which costs money and should be avoided.
i. Operator waits for the PS to handle the situation, does not act in time

UCA 6: Close MSIV provided when there is no rupture/leak

(1) Secondary cooling system
   a. Feedwater pumps not working properly
   b. Condenser leaking (loosing water)
   c. Too much sludge in water (blocking water)
   d. Object in water that could cut flux to SG
   e. Spurious opening of relief valves

(2) Process Feedback
   a. SG level feedback not provided
   b. SG Pressure low (setpoints not correct)
   c. Steam generator water level delayed or incorrect
   d. False SG isolation signal[37]
   e. Main steam line activity (false positive signal)

[37] This could occur, for example, in a situation where the water level at the SG is low concurrent with a SG low pressure, which could be due to a open Relief Valve.

f. Conflicting data indicating a false situation where close valve would be needed
g. Voting system does not work properly and gives wrong measures
h. Sensor Failure

(3) Outside Information

a. PZR pressure indication delayed
b. PZR feedback missing
c. False PZR pressure feedback
d. False feedback shows PZR level as low
e. False signal of initiation of SI
f. False Partial cool down initiated signal
g. Startup/shutdown not recognized[38]
h. Wrong combination of indicators from the 4 divisions

(4) Operator

a. Operator thinks Steam Generator Tubes are ruptured when they are not
b. Operator thinks the main steam line has a leak when it does not
c. Operator thinks main feedwater has a leak when it does not
d. Operator confused about the procedure to be followed
e. Operator confused because of conflicting indicators
f. Blank screen induces operator to think situation is different
g. False alarm of radiation
h. Close wrong valve, other SG

---

[38] One of the causes for wrong command can be confusion about indicators. The controllers check several indicators to decide what the specific problem is. For example, the main steam pipe break would also cause high pressure in the main steam line compartment. Or, SG low level combined with permissive 13 (startup) means there is no need to isolate the SG. It could happen that there would be a problem with the sensors, or the model (inside the controller) could be wrong, or algorithm could be wrong. “Confusion" could mean the model is not clear, or that there is an overlap of values.

A.8.1.2 Causal factors leading to an operator control action not being followed

In addition to identifying why unsafe control actions might be provided, it is important to analyze how safe control actions may not be followed appropriately. This section identifies how the safety constraints could be violated even if safe control actions are provided. Figure 43 shows areas of the control loop in which additional causal factors can lead to a violation of Safety Constraints 1 to 6.

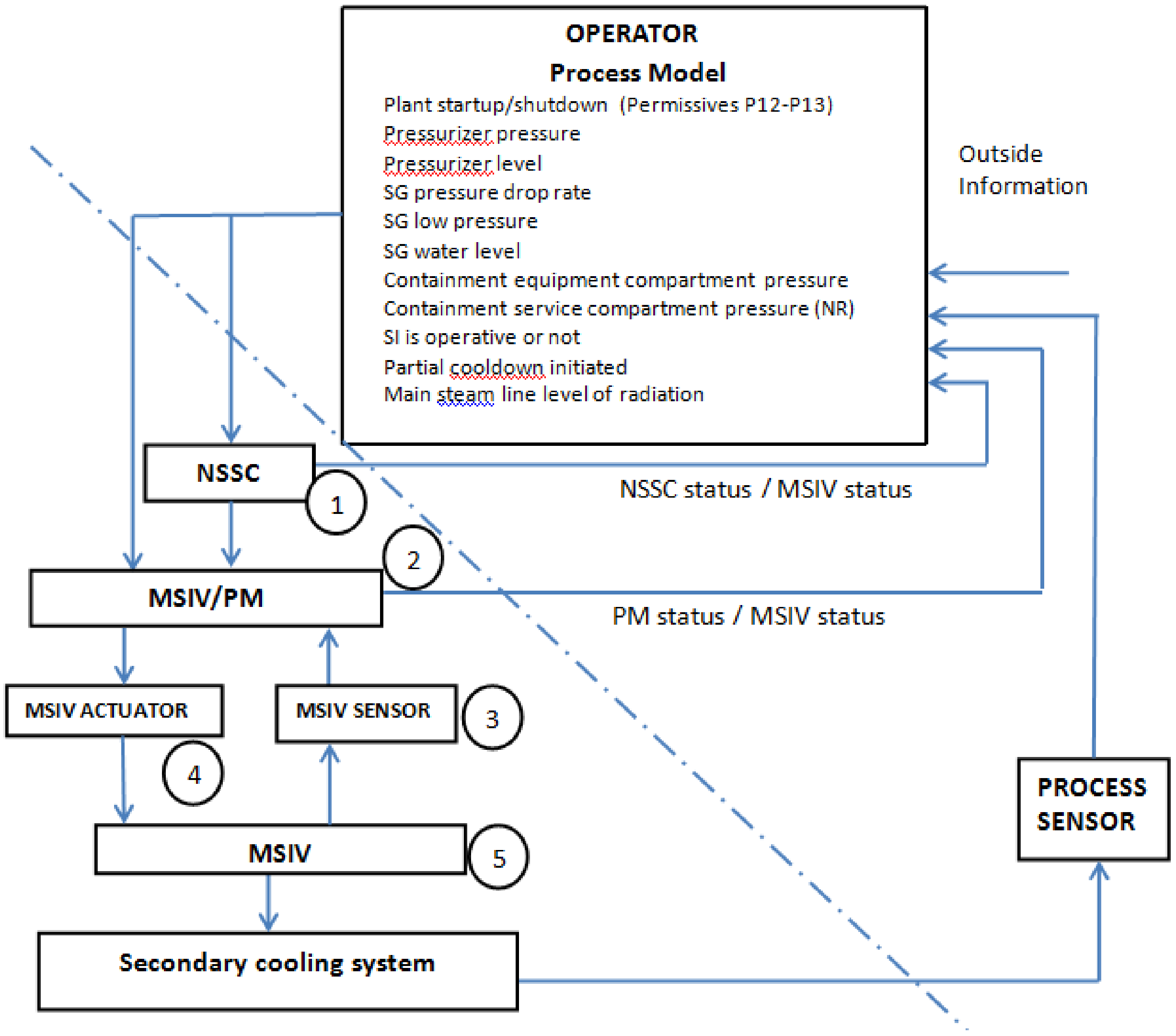


Figure 43: Causal factors leading to operator unsafe control actions not being followed

<u>SC 1</u>: MSIV must be closed when there is a leak (rupture in the SG tube, leak in main feedwater, or leak in main steam line) and the support systems are adequate.
<u>SC 2</u>: MSIV must be closed when there is a main feedwater or main steam line leak and other support systems are inadequate.

<u>Basic Scenario</u>: Operator provides *Close MSIV* command, but MSIV does NOT close

(1) NSSC
  a. Physical damage/failure
  b. Does not recognize operator command
  c. Manufacturing defects
  d. Inadequate algorithm
  e. Loss of power or blackout

(2) PM
  a. Wrong priority settings causing PM to ignore the close command
  b. Does not recognize PS or manual command
  c. Physical damage/failure
  d. Multiplex malfunctioning
  e. An operation (for example checking status of MSIV actuators) takes much longer time than expected/required, and PM ignores new commands
  f. Two conflicting commands come at the same or nearly the same time, from different controllers: the first one with lower priority than the second one.
  g. PM previously received interlock command from PS or other controller (e.g. to prevent MSIV closure during startup), causing PM to ignore operator commands to close MSIV
  h. Conflicting commands are sent (operator/PS, PS/DAS, etc.)
  i. Manufacturing defects
  j. Loss of power or blackout

(3) MSIV Sensor
  a. Reports device operational when it is not (therefore close command cannot be followed)

b. Reports valve position as open when it is not (therefore close command was sent but cannot be followed)
c. Physical damage/failure
d. Manufacturing defects
e. Loss of power or blackout

(4) MSIV Actuator

a. In the case of unavailability of the oil pump (lack of power supply) if the MSIV is already open, then it automatically remains open for a certain period of time.
b. Mechanical failure in the dump valves, preventing the oil from coming to the tank.
c. Debris prevents the valve to be closed, making it to remain partially or completely open
d. The nitrogen pressure, in the upper chamber, is not enough to close the valve, which had not been reported accordingly
e. Upper chamber is under maintenance to restore pressure
f. Dump valves do not open due to mechanical failures
g. Physical damage/failure
h. Manufacturing defects
i. Loss of power or blackout

(5) MSIV

a. The pressure in the lower chamber does not drop
b. The gate of the valve get stuck and does not move
c. Upper has very low pressure that creates a vacuum preventing the piston from moving
d. The upper chamber pressure is not enough to push the piston
e. Debris inside the valve prevent it from closing completely or partially
f. Physical damage/failure
g. Manufacturing defects

<u>Safety Constraints 3-6:</u>

<u>SC 3</u>: MSIV must not be closed when there is a SGTR and support systems are inadequate

<u>SC 4</u>: MSIV must not be closed too early while SG pressure is too high

<u>SC 5</u>: MSIV must not be closed too late after rupture/leak (in the SG tube, main feedwater, or main steam line)

<u>SC 6</u>: MSIV must not be closed when there is no rupture/leak

<u>Basic Scenario</u>: Operator does not provide *Close MSIV* command, but MSIV closes

(1) NSSC
  a. Physical damage/failure
  b. Some error in NSSC algorithm[39]
  c. NSSC has manufacturing defect
  d. Manufacturing defects
  e. Loss of power or blackout
  f. Inadequate algorithm

(2) PM
  a. PM holds execution of command requests due to interlock issued by PS. This causes delaying a new command
  b. Wrong priority settings (e.g. causing valve to close too late or too early)
  c. Does not recognize PS or manual command
  d. Physical damage/failure
  e. Multiplex malfunctioning
  f. Conflicting commands are sent (operator/PS, PS/DAS, etc.)[40]

[39] As the Operator has to follow a procedure to disable the NSSC automated control to enable manual control, it could happen that the NSSC, through some programming error, starts a control action after it is disabled, at the same time it is disabled, or starts a control action that it never received for some other reason.

[40] Conflicting commands may be sent, for example and operator command sent at the same time as a PS command, causing PM to lock up or execute the wrong command. There may also be problems due to DAS activation after

    g. Manufacturing defects
    h. Loss of power or blackout

(3) MSIV Sensor

    a. Reports device not operational when it is (therefore PM does not forward close command)
    b. Shows valve position as closed when it is open or only partially closed (therefore PM does not forward close command)
    c. Physical damage/failure
    d. Manufacturing defects

(4) MSIV Actuator

    a. The oil pump may have mechanical problems which causes the valve to automatically be kept open, causing delay
    b. The pilots are de-energized (two pilots in series), then the dump valve opens which closes the valve too early
    c. Mechanical failure in the dump valve
    d. Mechanical failure dumps the hydraulic oil from lower chamber and closes valve
    e. Test of closure causes it to be inadvertently closed
    f. Physical damage/failure
    g. Manufacturing defects
    h. Loss of power or blackout

(5) MSIV

    a. Leakage in the upper chamber makes pressure to be not enough to close the valve at the right time, hence delay
    b. A mismatch between the necessary pressure, in the oil chamber, to keep the valve open and the actual pressure applied, may cause that the oil pressure is not enough to keep it open causing it to close. Project mistake or assemblage mistake.

---

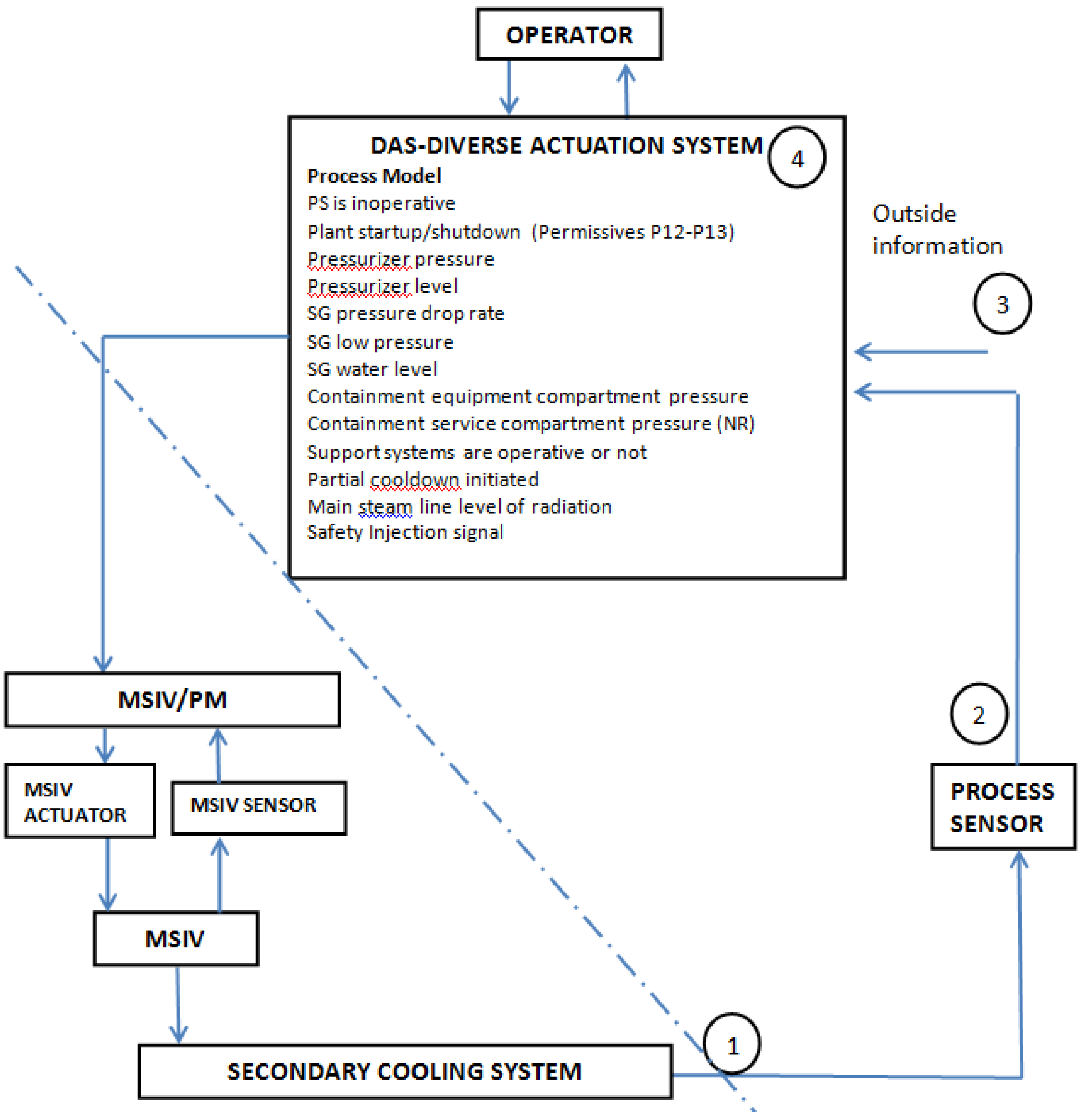
previous PS commands, or other commands sent before PM has finished executing them. Some commands may be ignored because PM ignores all commands until the current command is finished executing, even if it takes a fraction of a second.

c. A mismatch between the minimum pressure in the nitrogen chamber necessary to close the valve may cause that the pressure applied is higher than the necessary and this may cause the valve to be closed. Project mistake or an assemblage mistake.
d. Physical damage/failure
e. Manufacturing defects

A.8.2 DAS Causal Factors

A.8.2.1 Causal factors leading to DAS unsafe control actions

This section identifies causal factors that can lead to each unsafe control action summarized in Figure 29 for the DAS.



OPERATOR
DAS-DIVERSE ACTUATION SYSTEM
4
Process Model
PS is inoperative
Plant startup/shutdown (Permissives P12-P13)
Pressurizer pressure
Pressurizer level
SG pressure drop rate
SG low pressure
SG water level
Containment equipment compartment pressure
Containment service compartment pressure (NR)
Support systems are operative or not
Partial cooldown initiated
Main steam line level of radiation
Safety Injection signal
Outside information
3
MSIV/PM
MSIV ACTUATOR
MSIV SENSOR
MSIV
2
PROCESS SENSOR
1
SECONDARY COOLING SYSTEM

Figure 44: Causal factors leading to DAS unsafe control actions

UCA 1: Close MSIV not provided when there is a leak (rupture in the SG tube, leak in main feedwater, or leak in main steam line) and the support systems are adequate

(1) Secondary cooling system (CVCS or emergency feedwater system)
   a. A concurrent situation could mask another. For example, a feedwater problem could happen concurrently with SGTR, and the SG water level may stay practically stable.
   b. Event progresses too slowly to detect
   c. Actuation of CVCS could make up for loss of coolant inventory making DAS delay actuation.

(2) Process Feedback
   a. SG level feedback missing, delayed, or incorrect
   b. SG Pressure, or setpoints, not correct
   c. Steam generator water level delayed
   d. Main steam line activity not correctly indicated
   e. Conflicting data indicating a false situation
   f. Voting system does not work properly and gives wrong measures
   g. No indication of partial cool down initiated
   h. Sensor failure

(3) Outside information
   a. PZR pressure delayed
   b. PZR feedback missing
   c. False feedback indicates PZR level is normal
   d. No indication of SI initiated
   e. Delayed indication of SI initiated
   f. Permissives wrongly in effect[41]

[41] One of the causes for wrong command can be confusion about indicators. “Confusion" could mean the model is not clear, there is an overlap of responsibilities, or conflicting process values are indicated. The controllers check several indicators to decide what the specific problem is. For example, the main steam pipe break would also cause high pressure in the main steam line compartment and low SG water level. However, SG low level together with

g. Wrong combination of indicators from the 4 divisions

(4) DAS- Diverse Actuation System

a. DAS does not recognize Steam Generator as ruptured when it is ruptured
b. DAS does not recognize the main steam line has a leak
c. DAS does not recognize the main feedwater has a leak
d. DAS does not recognize that PS is malfunctioning or non-operational and does not take control
e. DAS has no power supplied
f. DAS follows incorrect algorithm
g. DAS has wrong process model
h. Physical damage/failure
i. Manufacturing defects
j. Loss of power or blackout

UCA 2: Close MSIV not provided when there is a main feedwater or main steam line leak and other support systems are inadequate

(1) Secondary cooling system (CVCS or emergency feedwater system)

a. A concurrent situation could mask another.
b. Event progresses too slowly to detect
c. Actuation of CVCS could make up for loss of coolant inventory making DAS delay actuation.

(2) Process Feedback

a. SG level feedback missing, delayed, or incorrect
b. SG Pressure, or setpoints, not correct
c. Steam generator water level delayed
d. Conflicting data indicating a false situation

---

permissive 13 (startup) may indicate there is no need to isolate the SG. It could happen that there is a problem with the sensors, the model (inside the controller) could be wrong, or the algorithm could be wrong

e. Voting system does not work properly and gives wrong measures
f. No indication of partial cool down initiated
g. Sensor failure

(3) Outside information

a. PZR pressure delayed
b. PZR feedback missing
c. False feedback indicates PZR level is normal
d. No indication of SI initiated
e. Delayed indication of SI initiated
f. Permissives wrongly in effect[42]
g. Wrong combination of indicators from the 4 divisions

(4) DAS- Diverse Actuation System

a. DAS does not recognize the main steam line has a leak
b. DAS does not recognize the main feedwater has a leak
c. DAS incorrectly believes problem is SGTR when there is actually a main steam line or main feedwater leak
d. DAS does not recognize that PS is malfunctioning or non-operational and does not take control
e. DAS has no power supplied
f. DAS follows incorrect algorithm
g. DAS has wrong process model
h. Physical damage/failure
i. Manufacturing defects
j. Loss of power or blackout

[42] One of the causes for wrong command can be confusion about indicators. “Confusion" could mean the model is not clear, there is an overlap of responsibilities, or conflicting process values are indicated. The controllers check several indicators to decide what the specific problem is. For example, the main steam pipe break would also cause high pressure in the main steam line compartment and low SG water level. However, SG low level together with permissive 13 (startup) may indicate there is no need to isolate the SG. It could happen that there is a problem with the sensors, the model (inside the controller) could be wrong, or the algorithm could be wrong

<u>UCA 3</u>: Close MSIV provided when there is a SGTR but support systems are inadequate

(1) Secondary cooling system
    a. A concurrent situation could mask another and other support systems could appear adequate but may not be. For example, suppose main steam line high radioactivity is detected coincident with safety injection. This may make the controller assume that a partial cooldown was initiated when it may not have. Closing the MSIV would cause the SG pressure to rise in this case.

(2) Process Feedback
    a. SG level feedback not provided
    b. SG Pressure not correct
    c. Steam generator water level not correct
    d. Conflicting data indicating a false situation
    e. Voting system does not work properly and gives wrong measures
    f. Sensor failure

(3) Outside information
    a. Wrong combination of indicators from the 4 divisions
    b. PZR pressure delayed or missing
    c. False signal SI initiated

(4) DAS - Diverse Actuation System
    a. DAS does not recognize that the support systems are not working due to conflicting information
    b. DAS incorrectly believes problem is main steam line leak or feedwater leak when it is actually SGTR
    c. DAS has an inadequate algorithm
    d. DAS close valve while other SG valves are under maintenance or by mistake
    e. Physical damage/failure
    f. Manufacturing defects

<u>UCA 4</u>: Close MSIV provided too early (while SG pressure is high)

(1) Secondary cooling system
   a. A concurrent situation could mask another. For example, a feedwater problem could happen concurrent with a SGTR, and the SG water level stay practically stable.
   b. Event progress too slowly to detect
   c. Actuation of CVCS could make up for loss of coolant inventory making DAS delay actuation.

(2) Process Feedback
   a. SG level feedback not provided
   b. SG Pressure, or setpoint, is not correct
   c. Steam generator water level delayed
   d. Main steam line activity not correctly indicated
   e. Conflicting data indicating a false situation
   f. Voting system does not work properly and gives wrong measures
   g. Sensor failure

(3) Outside Information
   a. PZR pressure delayed
   b. PZR feedback missing
   c. False feedback indicates PZR level is normal
   d. No indication of SI initiated
   e. No indication of partial cool down initiated
   f. Permissives wrongly in effect[43]
   g. Wrong combination of indicators from the 4 divisions

(4) DAS - Diverse Actuation System

[43] This could occur, for example, in a situation where the water level at the SG is low concurrent with a SG low pressure, which could be due to a open Relief Valve.

a. DAS has conflicting information indicating it is already safe to initiate action after indications confirm rupture/leak
b. Physical damage/failure
c. Manufacturing defects
d. DAS has an inadequate algorithm
e. DAS has wrong process model

<u>UCA 5</u>: Close MSIV command provided too late after rupture/leak (in the SG tube, main feedwater, or main steam line)

(1) Secondary cooling system
  a. A concurrent situation could mask another. For example, a feedwater problem could happen concurrently with a SGTR such that the SG water level stays practically stable.
  b. Event progress too slowly to detect
  c. Actuation of CVCS could make up for loss of coolant inventory making DAS delay actuation.

(2) Process Feedback
  a. SG level feedback not provided
  b. SG Pressure, or setpoint, is not correct
  c. Steam generator water level delayed
  d. Main steam line activity not correctly indicated
  e. Conflicting data indicating a false situation
  f. Voting system does not work properly and gives wrong measures
  g. Sensor failure

(3) Outside Information
  a. PZR pressure delayed
  b. PZR feedback missing
  c. False feedback indicates PZR level is normal
  d. No indication of SI initiated

e. No indication of partial cool down initiated
f. Permissives wrongly in effect
g. Wrong combination of indicators from the 4 divisions

(4) DAS - Diverse Actuation System

a. DAS does not recognizes the real situation until it is too late after SGTR
b. DAS does not recognizes the real situation until it is too late after the main steam line leak
c. DAS does not recognizes the real situation until it is too late after the main feedwater leak
d. DAS has an inadequate algorithm
e. DAS has wrong process model
f. Physical damage/failure
g. Manufacturing defects
h. Loss of power or blackout

UCA 6: Close MSIV provided when there is no rupture/leak

(1) Secondary cooling system

a. Feedwater pumps not working properly
b. Condenser leaking (loosing water)
c. Too much sludge in water (blocking water)
d. Object in water that could cut flux to SG
e. Spurious opening of relief valves

(2) Process Feedback

a. SG level feedback not provided
b. SG Pressure low (setpoints not correct)
c. Steam generator water level delayed

d. False SG isolation signal [44]
e. Main steam line activity (false positive signal)
f. Conflicting data indicating a false situation where close valve would be needed
g. Voting system does not work properly and gives wrong measures
h. Sensor failure

(3) Outside Information

a. PZR pressure delayed
b. PZR feedback missing
c. False PZR pressure
d. False feedback shows PZR level is low
e. False signal of initiation of SI
f. False partial cool down initiated signal
g. Startup/shutdown not recognized [45]
h. Wrong combination of indicators from the 4 divisions

(4) DAS - Diverse Actuation System

a. DAS has wrong information indicating Steam Generator tubes are ruptured when they are not
b. DAS has wrong information indicating that main steam line or main feedwater has leak when there is no leak
c. DAS has wrong process model
d. DAS has an inadequate algorithm
e. Physical damage/failure
f. Manufacturing defects

[44] This could occur, for example, in a situation where the water level at the SG is low concurrent with a SG low pressure, which could be due to a open Relief Valve.

[45] One of the causes for wrong command can be confusion about indicators. The controllers check several indicators to decide what the specific problem is. For example, the main steam pipe break would also cause high pressure in the main steam line compartment. Or, SG low level plus permissive 13 (startup) indicates no need to isolate the SG. It could happen that there would be a problem with the sensors, or the model (inside the controller) could be wrong, or algorithm could be wrong. “Confusion" could mean the model is not clear, or that there is an overlap of values

g. Loss of power or blackout

A.8.2.2 Causal factors leading to DAS control actions not being followed

This section identifies how the safety constraints could be violated even if safe control actions are provided. Figure 45 shows areas of the control loop in which additional causal factors can lead to a violation of Safety Constraints 1 to 6.

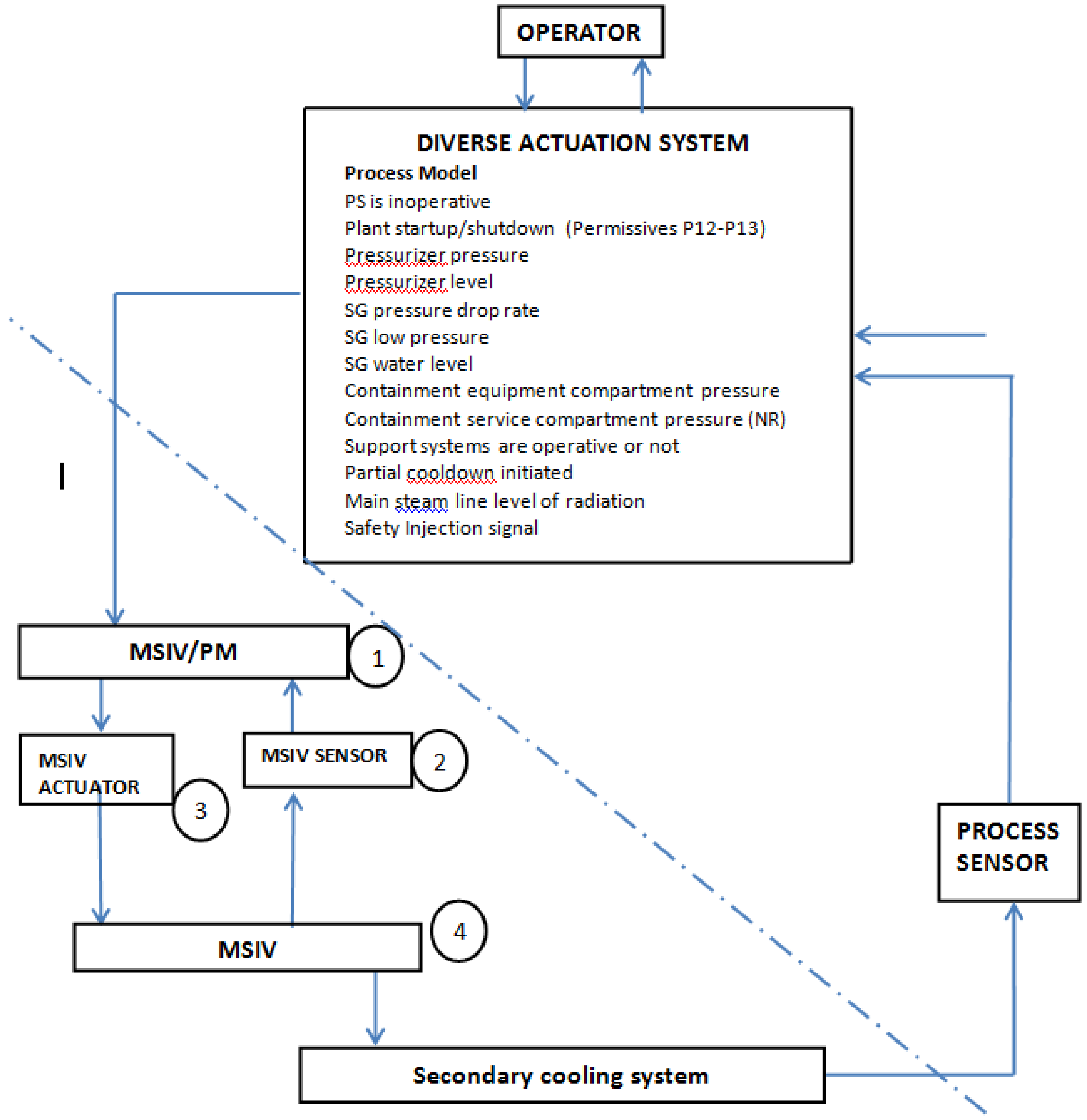


Figure 45: Causal factors leading to DAS control actions not being followed

<u>SC 1</u>: MSIV must be closed when there is a leak (rupture in the SG tube, leak in main feedwater, or leak in main steam line) and the support systems are adequate.
<u>SC 2</u>: MSIV must be closed when there is a main feedwater or main steam line leak and other support systems are inadequate.

<u>Basic Scenario</u>: DAS provides *Close MSIV* command, but MSIV does NOT close

(1) Priority Module
   a. Wrong priority settings causing PM to ignore the close command
   b. Does not recognize DAS command
   c. Physical damage/failure
   d. Multiplex malfunctioning
   e. Some operation (for example checking status of MSIV actuators) takes much longer time than supposed to, and PM ignores new commands
   f. Two conflicting action commands come at the same or nearly the same time, from different controllers: the first one with lower priority than the second one.
   g. PM had received a interlock command from PS, but PS goes down right after, so PM keeps waiting for new commands and does not accept new commands.
   h. Conflicting commands are sent (operator/PS, PS/DAS, etc.)
   i. Manufacturing defects
   j. Loss of power or blackout

(2) MSIV Sensor
   a. Reports device operational when it is not (therefore close command cannot be followed)
   b. Reports valve position as open when it is not (therefore close command was sent but cannot be followed)

(3) MSIV Actuator

a. In the case of unavailability of the oil pump (lack of power supply) if the MSIV is already open, then it automatically remains open for a certain period of time.
b. Mechanical failure in the dump valves, preventing the oil from coming to the tank.
c. Debris prevents the valve to be closed, making it to remain partially or completely open
d. The nitrogen pressure, in the upper chamber, is not enough to close the valve, which had not been reported accordingly
e. Upper chamber is under maintenance to restore pressure
f. Dump valves do not open due to mechanical failures
g. Physical damage/failure
h. Manufacturing defects
i. Loss of power or blackout

(4) MSIV Valve

a. Leakage in the upper chamber makes pressure to be not enough to close the valve at the right time, hence delay
b. A mismatch between the necessary pressure, in the oil chamber, to keep the valve open and the actual pressure applied, may cause that the oil pressure is not enough to keep it open causing it to close. Project mistake or assemblage mistake.
c. A mismatch between the minimum pressure in the nitrogen chamber necessary to close the valve may cause that the pressure applied is higher than the necessary and this may cause the valve to be closed. Project mistake or an assemblage mistake.
d. Physical damage/failure
e. Manufacturing defects

Safety Constraints 3-6:

SC 3: MSIV must not be closed when there is a SGTR and support systems are inadequate

SC 4: MSIV must not be closed too early while SG pressure is too high

SC 5: MSIV must not be closed too late after rupture/leak (in the SG tube, main feedwater, or main steam line)

SC 6: MSIV must not be closed when there is no rupture/leak

Basic Scenario: DAS does not provide *Close MSIV* command, but MSIV closes

(1) Priority Module

a. PM holds execution of command requests due to interlock issued by PS. This causes delaying a new command
b. PM receives close command from another cotnroller
c. Wrong priority settings
d. Does not recognize PS or manual command
e. Physical damage/failure
f. Multiplex malfunctioning
g. Conflicting commands are sent (operator/PS, PS/DAS, etc.)[46]
h. Physical damage/failure
i. Manufacturing defects
j. Loss of power or blackout

(2) MSIV Sensor

a. Reports device not operational when it is (therefore PM does not forward close command)

[46] Conflicting commands may be sent, for example and operator command sent at the same time as a PS command, causing PM to lock up or execute the wrong command. There may also be problems due to DAS activation after previous PS commands, or other commands sent before PM has finished executing them. Some commands may be ignored because PM ignores all commands until the current command is finished executing, even if it takes a fraction of a second.

b. Shows valve position as closed when it is open or only partially closed (therefore PM does not forward close command)
c. Physical damage/failure
d. Manufacturing defects

(3) MSIV Actuator

a. The oil pump may have mechanical problems which causes the valve to automatically be kept open, causing delay
b. The pilots are de-energized (two pilots in series), then the dump valve opens which closes the valve too early
c. Mechanical failure in the dump valve
d. Mechanical failure dumps the hydraulic oil from lower chamber and closes valve
e. Test of closure causes it to be inadvertently closed
f. Physical damage/failure
g. Manufacturing defects
h. Loss of power or blackout

(4) MSIV Valve

a. Leakage in the upper chamber makes pressure to be not enough to close the valve at the right time, hence delay
b. A mismatch between the necessary pressure, in the oil chamber, to keep the valve open and the actual pressure applied, may cause that the oil pressure is not enough to keep it open causing it to close. Project mistake or assemblage mistake.
c. A mismatch between the minimum pressure in the nitrogen chamber necessary to close the valve may cause that the pressure applied is higher than the necessary and this may cause the valve to be closed. Project mistake or an assemblage mistake.
d. Physical damage/failure
e. Manufacturing defects

A.8.3 PS Causal Factors

A.8.3.1 Causal factors leading to PS unsafe control actions

This section identifies causal factors that can lead to each unsafe control action summarized in Figure 29 for the PS.

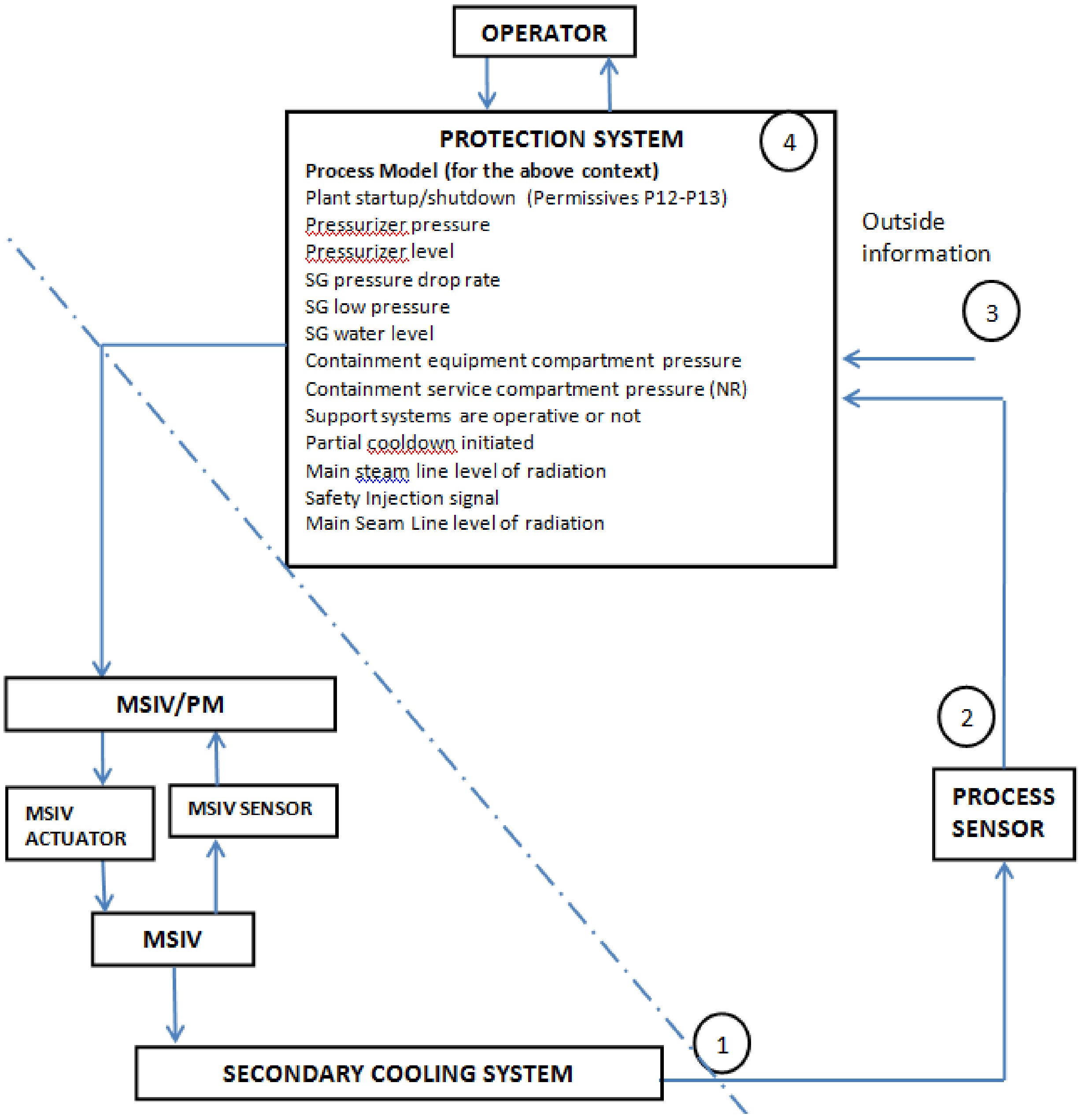

OPERATOR
PROTECTION SYSTEM
4
Process Model (for the above context)
Plant startup/shutdown (Permissives P12-P13)
Pressurizer pressure
Pressurizer level
SG pressure drop rate
SG low pressure
SG water level
Containment equipment compartment pressure
Containment service compartment pressure (NR)
Support systems are operative or not
Partial cooldown initiated
Main steam line level of radiation
Safety Injection signal
Main Seam Line level of radiation
Outside information
3
2
PROCESS SENSOR
MSIV/PM
MSIV ACTUATOR
MSIV SENSOR
MSIV
1
SECONDARY COOLING SYSTEM

Figure 46: Causal factors for PS unsafe control actions

UCA 1: Close MSIV not provided when there is a leak (rupture in the SG tube, leak in main feedwater, or leak in main steam line) and the support systems are adequate

(1) Secondary cooling system (CVCS or emergency feedwater system)
   a. A concurrent situation could mask another. For example, a feedwater problem could happen concurrently with a SGTR, and the SG water level may stay practically stable.
   b. Event progress too slowly to detect
   c. Actuation of CVCS could make up for loss of coolant inventory making PS delay actuation.

(2) Process Feedback
   a. SG level feedback missing, delayed, or incorrect
   b. SG Pressure, or setpoints, is not correct
   c. Steam generator water level delayed
   d. Main steam line activity not correctly indicated
   e. Conflicting data indicating a false situation
   f. Voting system does not work properly and gives wrong measures
   g. No indication of partial cool down initiated
   h. Sensor failure

(3) Outside information
   a. PZR pressure delayed
   b. PZR feedback missing
   c. False feedback indicates` PZR level is normal
   d. No indication of SI initiated
   e. Delayed indication of SI initiated
   f. Permissives wrongly in effect[47]

[47] One of the causes for wrong command can be confusion about indicators. "Confusion" could mean the model is not clear, there is an overlap of responsibilities, or conflicting process values are indicated. The controllers check several indicators to decide what the specific problem is. For example, the main steam pipe break would also cause high pressure in the main steam line compartment and low SG water level. However, SG low level together with

g. Wrong combination of indicators from the 4 divisions

(4) PS-Protection System

a. PS does not recognize Steam Generator is ruptured
b. PS does not recognize main steam line has a leak
c. PS does not recognize the main feedwater has a leak
d. PS has no power supply
e. PS follows inadequate algorithm
f. PS has a manufacturing defect
g. Physical damage/failure
h. Loss of power or blackout
i. PS has wrong process model

UCA 2: Close MSIV not provided when there is a main feedwater or main steam line leak and other support systems are inadequate

(1) Secondary cooling system (CVCS or emergency feedwater system)

a. A concurrent situation could mask another
b. Event progress too slowly to detect
c. Actuation of CVCS could make up for loss of coolant inventory making PS delay actuation.

(2) Process Feedback

a. SG level feedback missing, delayed, or incorrect
b. SG Pressure, or setpoints, is not correct
c. Steam generator water level delayed
d. Conflicting data indicating a false situation
e. Voting system does not work properly and gives wrong measures
f. No indication of partial cool down initiated
g. Sensor failure

---

permissive 13 (startup) may indicate there is no need to isolate the SG. It could happen that there is a problem with the sensors, the model (inside the controller) could be wrong, or the algorithm could be wrong.

(3) Outside information

   a. PZR pressure delayed
   b. PZR feedback missing
   c. False feedback indicates` PZR level is normal
   d. No indication of SI initiated
   e. Delayed indication of SI initiated
   f. Permissives wrongly in effect[48]
   g. Wrong combination of indicators from the 4 divisions

(4) PS-Protection System

   a. PS does not recognize main steam line has a leak
   b. PS does not recognize the main feedwater has a leak
   c. PS believes there is an SGTR when there is actually a main steam line or feedwater leak
   d. PS has no power supply
   e. PS follows inadequate algorithm
   f. PS has wrong process model
   g. PS has a manufacturing defect
   h. Physical damage/failure
   i. Loss of power or blackout

<u>UCA 3</u>: Close MSIV provided when there is a SGTR but support systems are inadequate

(1) Secondary cooling system

[48] One of the causes for wrong command can be confusion about indicators. “Confusion" could mean the model is not clear, there is an overlap of responsibilities, or conflicting process values are indicated. The controllers check several indicators to decide what the specific problem is. For example, the main steam pipe break would also cause high pressure in the main steam line compartment and low SG water level. However, SG low level together with permissive 13 (startup) may indicate there is no need to isolate the SG. It could happen that there is a problem with the sensors, the model (inside the controller) could be wrong, or the algorithm could be wrong.

    a. A concurrent situation could mask another and other support systems could appear adequate but may not be. For example, suppose main steam line high radioactivity is detected coincident with safety injection. This may make the controller assume that a partial cooldown was initiated when it may not have. Closing the MSIV would cause the SG pressure to rise in this case.

(2) Process Feedback

    a. SG level feedback not provided
    b. SG Pressure not correct
    c. Steam generator water level not correct
    d. Conflicting data indicating a false situation
    e. Voting system does not work properly and gives wrong measures
    f. Sensor failure

(3) Outside information

    a. Wrong combination of indicators from the 4 divisions
    b. PZR pressure delayed or missing
    c. False signal SI initiated

(4) PS-Protection System

    a. PS does not recognize that the support systems are not working due to conflicting information
    b. PS believes there is a main steam line or feedwater leak when there is actually an SGTR
    c. PS has an inadequate algorithm
    d. PS has wrong process model
    e. PS close valve while other SG valves are under maintenance or by mistake
    f. PS has a manufacturing defect
    g. Physical damage/failure
    h. Manufacturing defects
    i. Loss of power or blackout

UCA 4: Close MSIV provided too early (while SG pressure is high)

(1) Secondary cooling system

  a. A concurrent situation could mask another. For example, a feedwater problem could happen concurrent with a SGTR, and the SG water level stay practically stable.
  b. Event progresses too slowly to detect
  c. Actuation of CVCS could make up for loss of coolant inventory delaying PS actuation.

(2) Process Feedback

  a. SG level feedback not provided
  b. SG Pressure, or setpoints, not correct
  c. Steam generator water level delayed
  d. Main steam line activity not correctly indicated
  e. Conflicting data indicating a false situation
  f. Voting system does not work properly and gives wrong measures
  g. Sensor failure

(3) Outside Information

  a. PZR pressure delayed
  b. PZR feedback missing
  c. False feedback indicates PZR level is normal
  d. No indication of SI initiated
  e. No indication of partial cool down initiated
  f. Permissives wrongly in effect *
  g. Wrong combination of indicators from the 4 divisions

(4) PS-Protection System

  a. PS has an inadequate algorithm
  b. PS has conflicting information indicating it is already safe to initiate action after indications confirm rupture/leak
  c. Physical damage/failure
  d. Manufacturing defects
  e. Loss of power or blackout
  f. PS has wrong process model

<u>UCA 5</u>: Close MSIV provided too late after rupture/leak (in the SG tube, main feedwater, or main steam line)

(1) Secondary cooling system

   a. A concurrent situation could mask another. For example, a feedwater problem could happen concurrently with a SGTR such that the SG water level stays practically stable.
   b. Event progress too slowly to detect
   c. Actuation of CVCS could make up for loss of coolant inventory making PS delay actuation.

(2) Process Feedback

   a. SG level feedback not provided
   b. SG Pressure, or setpoints, is not correct
   c. Steam generator water level delayed
   d. Main steam line activity not correctly indicated
   e. Conflicting data indicating a false situation
   f. Voting system does not work properly and gives wrong measures
   g. Sensor failure

(3) Outside Information

   a. PZR pressure delayed
   b. PZR feedback missing
   c. False feedback indicates PZR level is normal
   d. No indication of SI initiated
   e. No indication of partial cool down initiated
   f. Permissives wrongly in effect
   g. Wrong combination of indicators from the 4 divisions

(4) PS-Protection System

   a. PS does not recognize the real situation until it is too late after SGTR
   b. PS does not recognize the real situation until it is too late after the main steam line or feedwater leak

c. PS has an inadequate algorithm
d. PS has wrong process model
e. PS has a manufacture defect
f. Physical damage/failure
g. Loss of power or blackout

UCA 6: Close MSIV provided when there is no rupture/leak

(1) Secondary cooling system
   a. Feedwater pumps not working properly
   b. Condenser leaking (loosing water)
   c. Too much sludge in water (blocking water)
   d. Object in water that could cut flux to SG
   e. Spurious opening of relief valves

(2) Process Feedback
   a. SG level feedback not provided
   b. SG Pressure low (setpoints not correct)
   c. Steam generator water level delayed
   d. False SG isolation signal[49]
   e. Main steam line activity (false positive signal)
   f. Conflicting data indicating a false situation where close valve would be needed
   g. Voting system does not work properly and gives wrong measures
   h. Sensor Failure

(3) Outside Information
   a. PZR pressure delayed
   b. PZR feedback missing

[49] This could occur, for example, in a situation where the water level at the SG is low concurrent with a SG low pressure, which could be due to a open Relief Valve.

c. False PZR pressure
d. False feedback indicates PZR level is low
e. False signal of initiation of SI
f. False Partial cool down initiated signal
g. Startup/shutdown not recognized [50]
h. Wrong combination of indicators from the 4 divisions

(4) PS-Protection System

a. PS has wrong information indicating Steam Generator tubes are ruptured when they are not
b. PS has wrong information indicating that main steam line or feedwater has a leak they do not
c. PS has wrong process model
d. PS has an inadequate algorithm
e. PS has a manufacture defect
f. Physical damage/failure
g. Loss of power or blackout

[50] One of the causes for wrong command can be confusion about indicators. The controllers check several indicators to decide what the specific problem is. For example, the main steam pipe break would also cause high pressure in the main steam line compartment. Or, SG low level together with permissive 13 (startup)indicates no need to isolate the SG. It could happen that there would be a problem with the sensors, or the model (inside the controller) could be wrong, or algorithm could be wrong. “Confusion" could mean the model is not clear, or that there is an overlap of values.

A.8.3.2 Causal factors leading to PS control actions not being followed

This section identifies how the safety constraints could be violated even if safe control actions are provided. Figure 47 shows areas of the control loop in which additional causal factors can lead to a violation of Safety Constraints 1 to 6.

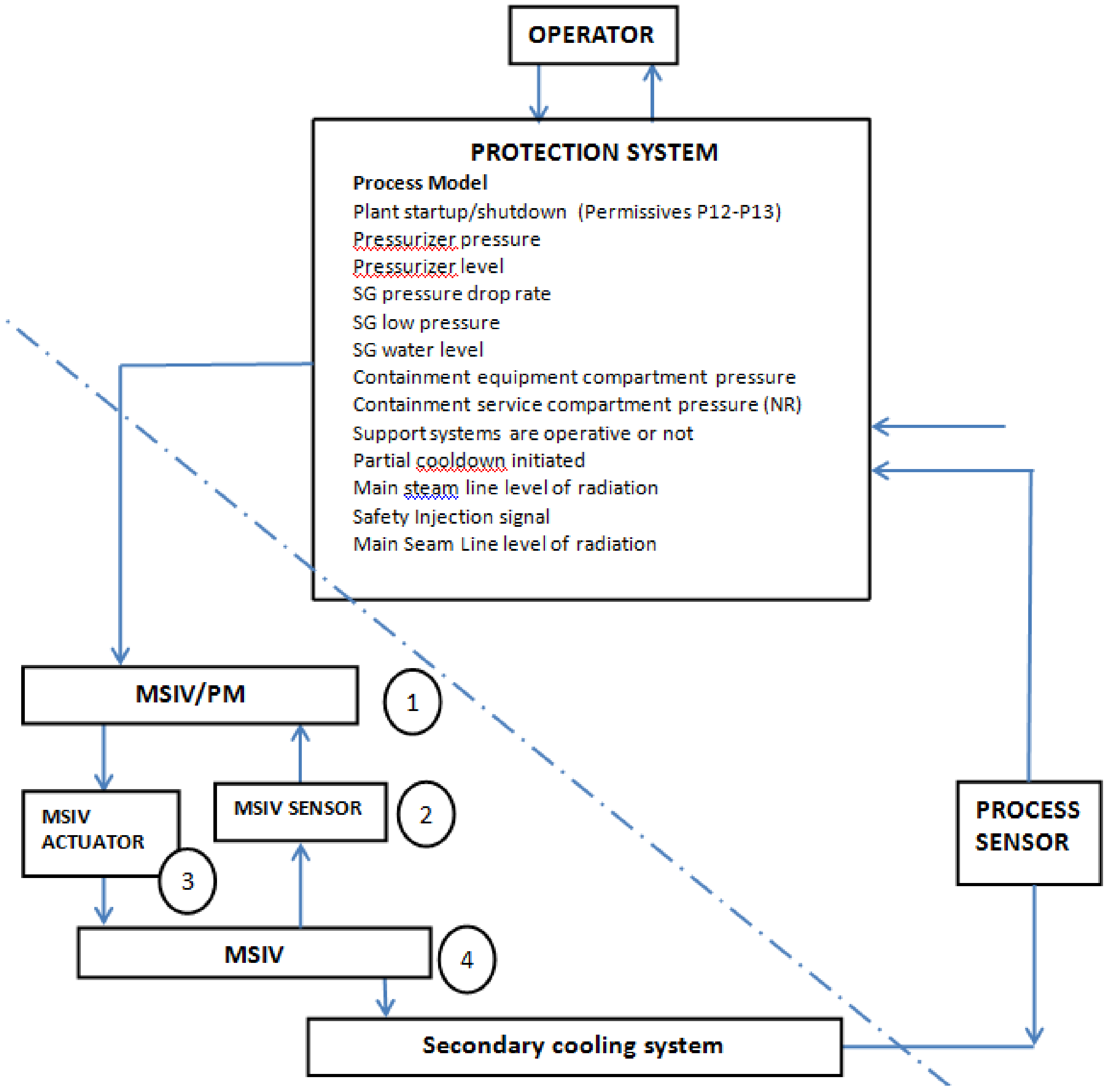

OPERATOR
PROTECTION SYSTEM
Process Model
Plant startup/shutdown (Permissives P12-P13)
Pressurizer pressure
Pressurizer level
SG pressure drop rate
SG low pressure
SG water level
Containment equipment compartment pressure
Containment service compartment pressure (NR)
Support systems are operative or not
Partial cooldown initiated
Main steam line level of radiation
Safety Injection signal
Main Seam Line level of radiation
MSIV/PM
1
MSIV ACTUATOR
MSIV SENSOR
2
3
MSIV
4
PROCESS SENSOR
Secondary cooling system

Figure 47: Causal factors leading to PS control actions not being followed

<u>SC 1</u>: MSIV must be closed when there is a leak (rupture in the SG tube, leak in main feedwater, or leak in main steam line) and the support systems are adequate.
<u>SC 2</u>: MSIV must be closed when there is a main feedwater or main steam line leak and other support systems are inadequate.

<u>Basic Scenario</u>: PS provides *Close MSIV* command, but MSIV does NOT close

(1) Priority Module
   a. Wrong priority settings causing PM to ignore the close command
   b. Does not recognize PS command
   c. Physical damage/failure
   d. Multiplex malfunctioning
   e. An operation (for example checking status of MSIV actuators) takes longer time than expected/required, and PM ignores new commands
   f. Two conflicting action commands come at the same or nearly the same time, from different controllers: the first one with lower priority than the second one.
   g. PM had received a interlock command from PS, which is not removed so PM does not accept new commands.
   h. Conflicting commands are sent (operator/PS, PS/DAS, etc.)
   i. Manufacturing defects
   j. Loss of power or blackout

(2) MSIV Sensor
   a. Reports device operational when it is not (therefore close command cannot be followed)
   b. Reports valve position as open when it is not (therefore close command was sent but cannot be followed)
   c. Physical damage/failure
   d. Manufacturing defects
   e. Loss of power or blackout

(3) MSIV Actuator

a. In the case of unavailability of the oil pump (lack of power supply) if the MSIV is already open, then it automatically remains open for a certain period of time.
b. Mechanical failure in the dump valves, preventing the oil from coming to the tank.
c. Debris prevents the valve to be closed, making it to remain partially or completely open
d. The nitrogen pressure, in the upper chamber, is not enough to close the valve, which had not been reported accordingly
e. Upper chamber is under maintenance to restore pressure
f. Dump valves do not open due to mechanical failures
g. Physical damage/failure
h. Manufacturing defects
i. Loss of power or blackout

(4) MSIV Valve

a. Leakage in the upper chamber makes pressure to be not enough to close the valve at the right time, hence delay
b. A mismatch between the necessary pressure, in the oil chamber, to keep the valve open and the actual pressure applied, may cause that the oil pressure is not enough to keep it open causing it to close. Project mistake or assemblage mistake.
c. A mismatch between the minimum pressure in the nitrogen chamber necessary to close the valve may cause that the pressure applied is higher than the necessary and this may cause the valve to be closed. Project mistake or an assemblage mistake.
d. Physical damage/failure
e. Manufacturing defects

<u>Safety Constraints 3-6:</u>

<u>SC 3</u>: MSIV must not be closed when there is a SGTR and support systems are inadequate

<u>SC 4</u>: MSIV must not be closed too early while SG pressure is too high

<u>SC 5</u>: MSIV must not be closed too late after rupture/leak (in the SG tube, main feedwater, or main steam line)

<u>SC 6</u>: MSIV must not be closed when there is no rupture/leak

<u>Basic Scenario</u>: PS does not provide *Close MSIV* command, but MSIV closes

(1) Priority Module
   a. PM holds execution of command requests due to interlock issued by PS. This causes delaying a new command
   b. Wrong priority settings
   c. Does not recognize PS or manual command
   d. Physical damage/failure
   e. Multiplex malfunctioning
   f. Conflicting commands are sent (operator/PS, PS/DAS, etc.)[51]
   g. Manufacturing defects
   h. Loss of power or blackout

(2) MSIV Sensor
   a. Reports device not operational when it is (therefore PM does not forward close command)
   b. Shows valve position as closed when it is open or only partially closed (therefore PM does not forward close command)
   c. Physical damage/failure

[51] Conflicting commands may be sent, for example and operator command sent at the same time as a PS command, causing PM to lock up or execute the wrong command. There may also be problems due to DAS activation after previous PS commands, or other commands sent before PM has finished executing them. Some commands may be ignored because PM ignores all commands until the current command is finished executing, even if it takes a fraction of a second.

    d. Manufacturing defects
    e. Loss of power or blackout

(3) MSIV Actuator

    a. The oil pump may have mechanical problems which causes the valve to automatically be kept open, causing delay
    b. The pilots are de-energized (two pilots in series), then the dump valve opens which closes the valve too early
    c. Mechanical failure in the dump valve
    d. Mechanical failure dumps the hydraulic oil from lower chamber and closes valve
    e. Test of closure causes it to be inadvertently closed
    f. Physical damage/failure
    g. Manufacturing defects
    h. Loss of power or blackout

(4) MSIV Valve

    a. Leakage in the upper chamber makes pressure to be not enough to close the valve at the right time, hence delay
    b. A mismatch between the necessary pressure, in the oil chamber, to keep the valve open and the actual pressure applied, may cause that the oil pressure is not enough to keep it open causing it to close. Project mistake or assemblage mistake.
    c. A mismatch between the minimum pressure in the nitrogen chamber necessary to close the valve may cause that the pressure applied is higher than the necessary and this may cause the valve to be closed. Project mistake or an assemblage mistake.
    d. Physical damage/failure
    e. Manufacturing defects

A.9 Extension to multiple steam generators

Thus far, the analysis has considered a single Steam Generator and a single MSIV. However, the results can be extended to multiple Steam Generators without repeating the entire analysis. One approach is to revise the existing context tables to reflect the control action “Close MSIV #1”. Because any feedwater or steamline leak will affect the control action “Close MSIV #1” in the same way as for the single SG system, these columns can remain the same. However, only a Steam Generator Tube Rupture in SG #1 is relevant to the closure of MSIV #1. Therefore, the values in the column Steam Generator Tube Rupture could be replaced with “SG #1 ruptured” and “SG #1 not ruptured”, while keeping the rest of the table the same. Similarly, the resulting table can then be converted for the other three MSIV commands by simply replacing #1 with #2, #3, or #4. If each redundant SG can compensate for the heat exchange performed by another SG then the definition of“other support systems” in both tables can be extended to include the other SGs.

A.10 Limitations of this analysis

This case study does not contain a detailed low-level analysis down to the individual components such as PLDs inside the PM. The small research grant, provided by the NRC, did not include the time or resources to analyze down to that level, and it was not the goal. STPA is a top-down analysis, and the analysis has been performed from the highest level (accidents and hazards) down to the module level to identify the control flaws that can cause hazards. The potential flaws and safety constraints found should be the starting point for a more detailed analysis. For example, it was found that the system-level design is such that incorrect priority settings for PM could cause a hazard if MSIV close commands are ignored. The next step would be to make sure that never happens. There are many options, including changing the system architecture (may not be practical at this point) or enforcing constraints on lower levels. The latter might be achieved by making the priority settings fixed within PM and not programmable and making sure PM internal logic and PLD design is such that MSIV commands are never ignored regardless of current priority. Other solutions are also possible. Of course, any potential solutions must be checked to ensure other safety constraints are not violated and new hazards are not introduced.

A.11 Results of the analysis

Although this study covered only a limited portion of the secondary cooling system, some important insights can be derived from it by examining the causes of unsafe control actions for the assumed scenarios.

An example insight obtained from the analysis is the difficulty of detecting a Steam Generator Tube Rupture (SGTR) through the normal indicators, which can lead to a delayed response by the automated controllers and the operator. The current solution relies on (i.e., gives credit to) the operator's ability to detect and intervene in certain cases. Relying on the operator, however, may not be effective because of other factors that will influence the operator decision-making process. These factors are identified in STPA Step 2 as possible causes for the operator not to provide the control action to close the MSIV or to provide it too late. The identified factors can be used to improve the design to make the operator error less likely or to mitigate it.

One reasonable recommendation, for example, is for regulators to ask the designers to simplify the indicators for the case of SGTR by making the level of radiation at the Main Steam Line a major indication to isolate the affected SG. This way, the Protection System (PS) would be able to detect the event earlier. In the current design, an indication of radioactivity is not sufficient for the PS to take action, and, as a result, there are additional scenarios in which neither the operator nor the PS may take action. For example, the operator may feel pressed to avoid spurious shutdowns and, as a consequence, he or she may wait longer for stronger evidence of the real problem. This type of response, in fact, is a common one identified by human factors experts in many real accidents. There could also be a situation where, after many years of work, the operator learns to completely rely on the automated controls to handle some incidents and becomes overconfident in its correct operation. This overreliance could lead to non-action or delayed action even though the analysis has assumed he or she will immediately take action in that case.

Part of the problem is the nuclear industry tendency to "credit the operator" (or credit some other device such as the PS), which means that the hazard analysis assumes that the operator (or other component) will detect and resolve the problem appropriately in a given situation. This thought process relates to the problem of only examining the "nominal" case versus identifying and resolving the worst case (as mentioned earlier in this dissertation). STAMP provides a more general alternative model that includes more potential paths (scenarios) to losses and can trace operator or other errors into the design to detect design flaws or weaknesses.

It is important to identify the factors under which a component, like the operator, may not act adequately and use those factors to improve the design of the system. The alternative is to simply blame the operators after an accident or incident for any failure to detect and resolve the problem as it was assumed they would. New NPP designs are placing the operators in a highly automated environment and telling them that the PS can handle almost everything. There are many subtle scenarios in which the PS may give up, or worse, ignore the problem without alerting the operator because it is assumed the operator will detect the problem and resolve it. Assuming that A is not safety-critical because B exists as a backup to A and that B is not safety-critical because it is only a backup system leads to circular reasoning and, potentially, accidents. A worst case analysis is necessary that assumes there may be design flaws or common-cause/common-mode failures in both.

The introduction of digital systems exacerbates the problem. Software allows highly complex systems to be created. While identifying safety-critical versus non-safety-critical components in a nuclear power plant was relatively straightforward for primarily electromechanical designs, the extensive use of software allows much more complex designs than previously possible and the potential for unintended and unexpected interactions among components. The more interactions between system components and the more complex the functional design, the more the opportunities for unintended effects and, consequently, the more opportunities for unsafe control actions that can lead to hazards. In other words, the more complex the system, the more possibilities of unintended effects due to the interactions among components. For example, the operator has to manually change settings by manipulating priority logic in order to allow NSSC to process the manual commands. This requirement can be a problem in case of an emergency.

Exhaustive system testing is not possible with software-intensive systems. Even if the individual components can be exhaustively tested, that will not guarantee system safety. The interactions between PM and other controllers and equipment are such that each component may operate in a reasonable manner given the local environment and information available, but from a global systems perspective the combined behavior of multiple components may be unsafe. For example, as discussed above, the PS may not take action in some situations where operator intervention is required while the operator may wait for the automated PS to take action. The STPA analysis in this case study was limited in scope to the MSIV commands and publically available information, but a more detailed STPA analysis seems warranted due to the central importance of this equipment in the control system.

Using a hazard analysis method based on STAMP allows more extensive analysis that includes events in which nothing failed but the hazards arise due to unsafe interactions among components. The identification of weaknesses in the overall PWR design are possible using STPA because the STPA analysis examines the interactions between the various controllers and system components. These weaknesses are unlikely to be found by hazard analysis methods based on assumptions about accidents being caused by chains of component failure events.

These are only some of the flaws or weaknesses in the design that can be identified from the partial system modeling and STPA hazard analysis performed for this research effort. A more complete modeling and analysis effort would most likely uncover even more.

A.12 Potential use of STPA in licensing

STAMP provides a more comprehensive basis for analyzing safety and licensing nuclear power plants. The following sections review several potential advantages.

A.12.1 Classification of components as safety-related vs. non-safety-related

While identifying safety-critical versus non-safety-critical components in a nuclear power plant was relatively straightforward for primarily electromechanical designs, the extensive use of software allows much more complex designs than previously possible and the potential for unintended and unexpected interactions among components. STPA does not begin with an assumption that certain equipment or controllers are safety-related and non-safety-related. Instead, an important output of STPA is a set of unsafe control actions for every controller analyzed and how they can directly or indirectly affect a hazard. The unsafe control actions identified in Step 1 describe how each controller can contribute to a hazardous situation. The output of STPA, therefore, could be used to classify components as safety-related or non-safety-related or to verify an existing classification. STPA Step 2 goes further and considers how each component—including sensors, actuators, logic devices, and communication paths—can contribute to hazardous situations. Analysts can then identify hazardous behavior related to the interactions between components that otherwise may not be captured by traditional analyses, and label each component appropriately.

Although there should be independence[52] between safety-related and non-safety-related controllers as classified in the U.S. EPR system, the STPA analysis on the example system in this case study showed that some systems classified as non-safety-related can still contribute to hazardous situations and are not truly independent from safety-related systems and functions. For example, NSSC, which is defined as a non-safety related controller, can hinder or slow down the successful closure of the MSIV when needed by reporting erroneous feedback to the operator or acting in unsafe or unexpected ways upon receiving a close MSIV command from the operator (or a combination of both). In this way, through its interaction with several safety-related controllers, NSSC can affect their ability to perform their safety-related functions.

As another example, the safety-related PM contains the non-safety-related communication device Profibus, which communicates with NSSC. Incorrect behavior of NSSC together with Profibus can potentially affect the safety-related functions of PM by potentially directly interfering with the control actions processed by PM. The interference could also be caused indirectly by interfering with the feedback provided to the operator or by providing inadequate or incorrect feedback to the operator. Without appropriate feedback, the operator cannot be assumed to be able to provide safe control actions, including MSIV and other controls.

[52] We use “independence” here as used in NUREG-0800: “data communication between safety channels or between safety and non-safety systems should not inhibit the performance of the safety function. … In practical terms, this means that for communications between safety and non-safety systems, the communications must be such that the safety system does not require any non-safety input to perform its safety function, and that any failure of the non-safety system, communications system, or data transmitted by the non-safety system will not prevent or influence that independent safety determination.” [NUREG-0800 Appendix 7.1-D].

A.12.2 Identifying potential operator errors and their causes and safety culture flaws

STAMP/STPA treats the operator as integral part of the system and thus an integral part of the hazard analysis. Factors such as “pressure to save time and money” can be as dangerous as a mechanical failure of a component and can be captured in this method.

A.12.3 Broadening the Analysis and Oversight

Other aspects of the overall socio-technical system can also be included in the STPA analysis although they were not included in the case study. The NRC has responsibility for overseeing safety culture and other aspects of nuclear power plant operations. The inclusion of social, organizational, and managerial factors in the hazard analysis (which is possible for STPA) can identify potential risks and leading indicators of increasing risk that the regulators can use to audit performance by the utilities.

A.12.4 Assisting in Understanding Applicant Functional Designs

The model of the safety control structure constructed as part of the STPA analysis can help regulatory authorities improve their understanding of the functional design of the system and can aid in communication and interchanges with applicants. In performing the case study, it was found that existing documentation provided a comprehensive description of the physical design, but there was great difficulty extracting the functional or logical design from this documentation. The control structure diagrams can help in providing this information and identifying missing information or ambiguous design descriptions.

The documentation for STPA can also facilitate discussions between experts from different disciplines, which in practice tend to speak different technical languages and have different perspectives and priorities. Simply using a control structure model of the system can help with communication among diverse groups concerning the functionality provided by the system design.

A.12.5 Enhancing the Review of Candidate Designs

STAMP/STPA can be used as a platform to provide the authorities with a broader and more systemic view of the system and can uncover unanticipated or unexpected behavior that emerges from the complex interactions that occur. This approach, as mentioned earlier, has the advantage of being able to capture both human and equipment behavior in the same control-theoretic model. Because the system is modeled in an integrated control structure rather than considering components in isolation, authorities may be better able to visualize weaknesses that otherwise would not be possible.

The Step 1 tables can provide a wide range of scenarios that could lead to unsafe control actions related to the identified hazards. These tables consider the possibilities of occurrences without relying on the availability or accuracy of probabilistic estimates, which makes STAMP/STPA a very powerful tool to assist in certification and licensing. Each unsafe control action can be directly and easily translated into component-level safety constraints, which can be compared with the safety requirements of an existing design to identify gaps, inconsistencies, or incompleteness. The Step 2 analysis guides the identification of possible causes of the unsafe control actions as well as other ways the safety constraints can potentially be violated. These results can also be used as a guide for the authorities to generate a list of requirements or mitigation measures that the licensee has to meet. Finally, the results can also be used as a basis to generate other requirements not yet identified, as there is a possibility that new issues will be raised after experts study the Step 1 and Step 2 results.

References

1. Vesely, W.E. and N.H. Roberts, *Fault Tree Handbook*. 1981: US Independent Agencies and Commissions.

2. Dekker, S., *Ten questions about human error : a new view of human factors and system safety*. Human factors in transportation. 2005, Mahwah, N.J.: Lawrence Erlbaum Associates. xix, 230 p.

3. Dekker, S., *The field guide to understanding human error*. 2006, Aldershot, England ; Burlington, VT: Ashgate. xv, 236 p.

4. Lutz, R.R. *Analyzing software requirements errors in safety-critical, embedded systems*. in *IEEE International Conference on Software Requirements*. 1992.

5. Leveson, N., *SafeWare : system safety and computers*. 1995, Reading, Mass.: Addison-Wesley. xvii, 680 p.

6. Bainbridge, L., *Ironies of Automation*, in *New Technology and Human Error*, J. Rasmussen, K. Duncan, and J. Leplat, Editors. 1987, John wiley and Sons: New York.

7. Sarter, N.B. and D.D. Woods, *How in the World Did We Ever Get into That Mode - Mode Error and Awareness in Supervisory Control.* Human Factors, 1995. 37(1): p. 5-19.

8. Hoffman, R.R. and L.G. Militello, *Perspectives on Cognitive Task Analysis: Historical Origins and Modern Communities of Practice*. 2012: Taylor & Francis.

9. Clarke, E.M., O. Grumberg, and D.A. Peled, *Model Cheking*. 1999: Mit Press.

10. Leveson, N., *Engineering a safer world : systems thinking applied to safety*. Engineering systems. 2012, Cambridge, Mass.: MIT Press.

11. Heinrich, H.W., *Industrial accident prevention: a scientific approach*. 1931: McGraw-Hill Book Company, inc.

12. Statistics, U.S.B.o.L., *Standardization of industrial accident statistics*. Bulletin of the United States Bureau of Labor Statistics. 1920: U.S. Dept. of Labor, Bureau of Labor Statistics.

13. Mitchell, J., M. Ramsey, and C. Dawson,*U.S. Blames Drivers, Not Toyota*, in *Wall Street Journal* 2011.

14. Kim, C.-R. and J. Crawley,*Toyota blames driver error for some unwanted speeding*, in *Reuters* 2010.

15. Waraich, O.,*BP Blames Halliburton for Gulf Oil Spill*, in *The Christian Post* 2011.

16. Bergin, T. and A. Rascoe,*BP and partners trade blame for oil spill*, in *Reuters* 2010.

17. Broder, J.M.,*Halliburton Rejects Blame for BP Cement Job*, in *New York times* 2010.

18. Broder, J.,*BP Shortcuts Led to Gulf Oil Spill, Report Says*, in *New York Times* 2011.

19. Bryant, B.,*Deepwater Horizon and the Gulf oil spill - the key questions answered*, in *The Guardian* 2011.

20. Fielder, J.H. and D. Birsch, *The Dc-10 Case: A Study in Applied Ethics, Technology, and Society*. Suny Series, Case Studies in Applied Ethics, Technology, and Society. 1992: State University of New York Press.

21. National Transportation Safety Board, *Aircraft Accident Report: American Airlines, Inc. McDonnell Douglas DC-10-10, N103AA. Near Windsor, Ontario, Canada*, 1973.

22. Branch, U.A.A.I., *Report on the accident in the Ermenonville Forest, France on March 3, 1973*, 1976.

23. Rasmussen, J., *Risk management in a dynamic society: a modelling problem.* Safety science, 1997. 27(2): p. 183-213.

24. Rasmussen, J., K. Duncan, and J. Leplat, *New technology and human error*. New technologies and work. 1987: J. Wiley.

25. Bird, F.E. and R.G. Loftus, *Loss control management*. 1976: Institute Press.

26. Adams, E., *Accident causation and the management system.* Professional Safety, 1976. 21(10): p. 26-29.

27. Reason, J., *Human Error*. 1990: Cambridge University Press.

28. Nance, J.,*Just How Secure is Airline Security?* , in *ABC News* 2005.

29. Perneger, T.V., *The Swiss cheese model of safety incidents: are there holes in the metaphor?* BMC health services research, 2005. 5(1): p. 71.

30. Reason, J., E. Hollnagel, and J. Paries, *Revisiting the «Swiss Cheese» Model of Accidents.* Journal of Clinical Engineering, 2006. 27: p. 110-115.

31. Hollnagel, E., D.D. Woods, and N. Leveson, *Resilience engineering: Concepts and precepts*. 2006: Ashgate Publishing Company.

32. Leveson, N.G., *Model-based analysis of socio-technical risk.* Massachusetts Institute of Technology, Cambridge, MA, USA, Tech. Rep. ESD-WP-2004-08, 2004.

33. Firesmith, D.G. *Engineering safety-related requirements for software-intensive systems*. in *Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on*. 2005.

34. Squair, M.J. *Issues in the application of software safety standards*. Australian Computer Society, Inc.

35. Zhe, C. and G. Motet. *Modeling System Safety Requirements Using Input/Output Constraint Meta-automata*. in *Systems, 2009. ICONS '09. Fourth International Conference on*. 2009.

36. Hollnagel, E. and O. Goteman, *The functional resonance accident model.* Cognitive System Engineering in Process Plant 2004.

37. McDonnell, M.D. and D. Abbott, *What is stochastic resonance? Definitions, misconceptions, debates, and its relevance to biology.* PLoS computational biology, 2009. 5(5): p. e1000348.

38. Benzi, R., A. Sutera, and A. Vulpiani, *The mechanism of stochastic resonance.* Journal of Physics A: mathematical and general, 1999. 14(11): p. L453.

39. Benzi, R., et al., *Stochastic resonance in climatic change.* Tellus, 1982. 34(1): p. 10-16.

40. Bulsara, A., et al., *Stochastic resonance in a single neuron model: Theory and analog simulation.* Journal of Theoretical Biology, 1991. 152(4): p. 531-555.

41. Bulsara, A.R. and F.E. Moss. *Single neuron dynamics: Noise-enhanced signal processing*. 1991. IEEE.

42. Cordo, P., et al., *Noise in human muscle spindles.* 1996.

43. Anishchenko, V.S., M.A. Safonova, and L.O. Chua, *Stochastic resonance in Chua's circuit driven by amplitude or frequency modulated signals.* International Journal of Bifurcation and Chaos, 1994. 4(02): p. 441-446.

44. McNamara, B., K. Wiesenfeld, and R. Roy, *Observation of stochastic resonance in a ring laser.* Physical Review Letters, 1988. 60(25): p. 2626-2629.

45. Iannelli, J.M., et al., *Stochastic resonance in a semiconductor distributed feedback laser.* Applied physics letters, 1994. 65(16): p. 1983-1985.

46. Blarer, A. and M. Doebeli, *Resonance effects and outbreaks in ecological time series.* Ecology Letters, 1999. 2(3): p. 167-177.

47. Xiao-Ming, M., S. Kai, and O. Qi, *Stochastic resonance in a financial model.* Chinese Physics, 2002. 11(11): p. 1106.

48. Wallace, R., D. Wallace, and H. Andrews, *AIDS, tuberculosis, violent crime, and low birthweight in eight US metropolitan areas: public policy, stochastic resonance, and the regional diffusion of inner-city markers.* Environment and Planning A, 1997. 29: p. 525-555.

49. Von Neumann, J. and O. Morgenstern, *Theory of Games and Economic Behavior (Commemorative Edition)*. 2007: Princeton university press.

50. Hollnagel, E., *The changing nature of risk.* Ergonomics Australia Journal, 2008. 22(1-2): p. 33-46.

51. Hollnagel, E., et al. *Analysis of Comair flight 5191 with the functional resonance accident model*. in *Proceedings of the 8th International Symposium of the Australian Aviation Psychology Association*. 2008.

52. Leveson, N., *A new accident model for engineering safer systems.* Safety science, 2004. 42(4): p. 237-270.

53. Leveson, N. *A systems model of accidents*. 2002. International Systems Safety Society: Unionville, USA.

54. Leveson, N.G., *Role of software in spacecraft accidents.* Journal of spacecraft and Rockets, 2004. 41(4): p. 564-575.

55. Watson, H., *Launch Control Safety Study*, 1961, Bell Laboratories: Murray Hill, NJ.

56. Ericson, C.A. *Fault Tree Analysis – A History*. in *Proceedings of The 17th International System Safety Conference*. 1999.

57. Bedford, T. and R. Cooke, *Probabilistic risk analysis: foundations and methods*. 2001: Cambridge University Press.

58. Ferson, S., *RAMAS Risk Calc 4.0 software: risk assessment with uncertain numbers*. 2002: CRC.

59. Kaiser, B., P. Liggesmeyer, and O. Mackel, *A new component concept for fault trees*, in *Proceedings of the 8th Australian workshop on Safety critical systems and software - Volume 33*2003, Australian Computer Society, Inc.: Canberra, Australia. p. 37-46.

60. Mannan, S. and F.P. Lees, *Lee's Loss Prevention in the Process Industries: Hazard Identification, Assessment, and Control*. 2005: Elsevier Butterworth-Heinemann.

61. Vaughan, D., *The Challenger launch decision : risky technology, culture, and deviance at NASA*. 1996, Chicago: University of Chicago Press. xv, 575 p.

62. Leveson, N.G., S.S. Cha, and T.J. Shimeall, *Safety verification of Ada programs using software fault trees.* Software, IEEE, 1991. 8(4): p. 48-59.

63. Leveson, N.G. and P.R. Harvey, *Software fault tree analysis.* Journal of Systems and Software, 1983. 3(2): p. 173-181.

64. Leveson, N. and C.S. Turner, *An Investigation of the Therac-25 Accidents.* Computer, 1993. 26(7): p. 18-41.

65. Lebron, J.E., *System safety study of minimum TCAS II*, 1983, Federal Aviation Administration: Washington, D.C. p. 360 p. in various pagings.

66. RTCA, *DO-312: Safety, Performance and Interoperability Requirements Document for the In-Trail Procedure in the Oceanic Airspace (ATSA-ITP) Application*, 2008: Washington, DC.

67. Childs, C., *Cosmetic System Safety.* Hazard Prevention, 1979.

68. Hixenbaugh, A.F., *Fault Tree for Safety*, 1968, Defense Technical Information Center, Boeing Co. Support Systems Engineering Seattle, WA Ft. Belvoir.

69. Dougherty, E.M., *Human Reliability-Analysis - Where Shouldst Thou Turn.* Reliability Engineering & System Safety, 1990. 29(3): p. 283-299.

70. Swain, A.D., *Human Reliability-Analysis - Need, Status, Trends and Limitations.* Reliability Engineering & System Safety, 1990. 29(3): p. 301-313.

71. RTCA and EUROCAE, *Guidelines for Approval of the Provision and Use of Air Traffic Services Supported by Data Communications*, 2002.

72. Moray, N., *Dougherty's dilemma and the one-sidedness of human reliability analysis (HRA).* Reliability Engineering &amp; System Safety, 1990. 29(3): p. 337-344.

73. Kantowitz, B.H. and Y. Fujita, *Cognitive theory, identifiability and human reliability analysis (HRA).* Reliability Engineering &amp; System Safety, 1990. 29(3): p. 317-328.

74. Mosleh, A., V.M. Bier, and G. Apostolakis, *A Critique of Current Practice for the Use of Expert Opinions in Probabilistic Risk Assessment.* Reliability Engineering & System Safety, 1988. 20(1): p. 63-85.

75. Ericson, C.A., *Hazard analysis techniques for system safety*. 2005, Hoboken, NJ: Wiley-Interscience. xx, 499 p.

76. Rasmussen, N.C., *Reactor safety study : an assessment of accident risks in U.S. commercial nuclear power plants*. 1975, Washington; Springfield, Va.: N.R.C.; National Technical Information Service.

77. Rausand, M. and A. Høyland, *System reliability theory : models, statistical methods, and applications*. 2nd ed. Wiley series in probability and statistics Applied probability and statistics. 2004, Hoboken, NJ: Wiley-Interscience. xix, 636 p.

78. Verma, A.K., *Reliability and safety engineering*. 1st ed. Springer series in reliability engineering. 2010, New York: Springer.

79. Skelton, B., *Process safety analysis : an introduction*. 1997, Houston, Tex.: Gulf Pub. xii, 213 p.

80. Apostolakis, G., et al.,*Integration of Reactor Design, Operations, and Safety*, in *MIT Course 22.39* 2006.

81. Acosta, C. and N. Siu, *Dynamic event trees in accident sequence analysis: application to steam generator tube rupture*. Reliability Engineering & System Safety, 1993. 41(2): p. 135-154.

82. Apostolakis, G. and T.L. Chu, *Time-dependent accident sequences including human actions*. Nucl. Technol.;(United States), 1984. 64(2).

83. Huang, D., T. Chen, and M.-J.J. Wang, *A fuzzy set approach for event tree analysis*. Fuzzy Sets and Systems, 2001. 118(1): p. 153-165.

84. Barriere, M., et al., *Technical basis and implementation guidelines for a technique for human event analysis (ATHEANA)*. NUREG-1624, Rev, 2000. 1.

85. Kletz, T.A., *Hazop and Hazan*. 2006: The Institution of Chemical Engineers.

86. Asnar, Y. and P. Giorgini, *Modelling Risk and Identifying Countermeasure in Organizations*, in *Critical Information Infrastructures Security*, J. Lopez, Editor. 2006, Springer Berlin Heidelberg. p. 55-66.

87. Walker, J.S., *Three Mile Island : a nuclear crisis in historical perspective*. 2004, Berkeley: University of California Press. xi, 303 p.

88. Marais, K., J.H. Saleh, and N.G. Leveson, *Archetypes for organizational safety*. Safety science, 2006. 44(7): p. 565-582.

89. Military, U.S., *MIL-P-1629 Procedures for Performing a Failure Modes Effects and Criticality Analysis*, 1949.

90. Dhillon, B.S., *Maintainability, maintenance, and reliability for engineers*. 2006, Boca Raton: CRC/Taylor & Francis. 217 p.

91. Stamatis, D.H., *Design for Six Sigma*. Six sigma and beyond. 2002: ST LUCIE PR.

92. JPL, N., *Failure Modes, Effects and Criticality Analysis (FMECA)*.

93. SAE, *ARP926 Design Analysis Procedure For Failure Modes, Effects and Criticality Analysis* 1967.

94. National Research Council, *Safety and offshore oil*. 1981: National Academy Press.

95. National Academy of Engineering, *Outer continental shelf resource development safety: a review of technology and regulation for the systematic minimization of environmental intrusion from petroleum products*. 1972: National Academy of Engineering, Marine Board.

96. Duckworth, H.A. and R.A. Moore, *Social Responsibility: Failure Mode Effects and Analysis*. Industrial Innovation Series. 2010: Taylor and Francis.

97. Crow, D. and A. Feinberg, *Design for Reliability*. Electronics Handbook Series. 2001: CRC Press.

98. Stamatis, D.H., *Failure Mode and Effect Analysis: Fmea from Theory to Execution*. 2003: ASQ Quality Press.

99. Goble, W.M., *Control systems safety evaluation and reliability*. 3rd ed. ISA resources for measurement and control series. 2010, Research Triangle Park, NC: International Society of Automation. xvi, 458 p.

100. Fullwood, R.R. and R.R. Fullwood, *Probabilistic safety assessment in the chemical and nuclear industries*. 2000, Boston: Butterworth-Heinemann. xxix, 514 p.

101. BP, *Deepwater Horizon Accident investigation Report.* 2010.

102. National Commission on the BP Deepwater Horizon Oil Spill and Offshore Drilling, *Deep Water: The Gulf Oil Disaster and the Future of Offshore Drilling.* 2010.

103. Lutz, R.R., *Analyzing software requirements errors in safety-critical, embedded systems*. Technical report. Iowa State University. Dept. of Computer Science. 1993, Ames, Iowa: Iowa State University Dept. of Computer Science. 14 p.

104. Leveson, N., *Completeness in formal specification language design for process-control systems*, in *Proceedings of the third workshop on Formal methods in software practice*2000, ACM: Portland, Oregon, United States. p. 75-87.

105. Panel, B., *The Report of the BP U.S. Refineries Independent Safety Review Panel.* 2007.

106. U.S. District Court for the Southern District of Texas Houston Division, *Statement of Facts.* 2007.

107. Lawley, H.G., *Operability studies and hazard analysis.* Chemical Engineering Progress, 1974. 70(4): p. 45-56.

108. Crawley, F., M. Preston, and B. Tyler, *Hazop: Guide to Best Practice: Guidelines to Best Practice for the Process and Chemical Industries*. 2008: Inst of Chemical Engineers.

109. Kletz, T.A., *Lessons from Disaster*. 1993, Houston: Gulf Publishing Company.

110. Dunjó, J., et al., *Hazard and operability (HAZOP) analysis. A literature review.* Journal of Hazardous Materials, 2010. 173(1–3): p. 19-32.

111. Fields, B. and P. Wright, *Error tolerance in collaborative systems.* 1988.

112. Burns, D.J. and R.M. Pitblado, *A modified HAZOP methodology for safety critical system assessment*. 1993: Springer Verlag.

113. Chudleigh, M.F. and J.N. Clare. *The benefits of SUSI: Safety analysis of user system interaction*. 1993. DTIC Document.

114. Cagno, E., F. Caron, and M. Mancini, *Risk analysis in plant commissioning: the Multilevel Hazop.* Reliability Engineering & System Safety, 2002. 77(3): p. 309-323.

115. Stringfellow, M., *Accident Analysis and Hazard Analysis for Human and Organizational Factors*, in *Aeronautics and Astronautics* 2010, MIT.

116. McDermid, J.A., et al. *Experience with the application of HAZOP to computer-based systems*. 1995. IEEE.

117. Glossop, M., A. Loannides, and J. Gould, *Review of hazard identification techniques*, 2000, Sheffield, UK: Health and Safety Laboratory.

118. Kletz, T.A., et al., *Computer Control & Human Error*. 1995: Inst of Chemical Engineers.

119. Fink, R., et al., *Data Management in Clinical Laboratory Information Systems*, in *Directions in Safety-Critical Systems*, F. Redmill and T. Anderson, Editors. 1993, Springer London. p. 84-95.

120. Andow, P., H. Great Britain, and E. Safety, *Guidance on Hazop Procedures for Computer-Controlled Plants*. HSE Contract Research Report. 1991: Stationery Office.

121. Lear, J., *Computer hazard and operability studies*, in *Sydney University Chemical Engineering Association Symposium: Safety and REliability of Process Control Systems*October 1993.

122. Nimmo, I., *Extend HAZOP to computer control systems.* Chemical Engineering Progress, 1994. 90(10): p. 32-44.

123. Hulin, B. and R. Tschachtli. *Identifying Software Hazards with a Modified CHAZOP*.

124. Reese, J.D. and N.G. Leveson. *Software Deviation Analysis*. in *Software Engineering, 1997., Proceedings of the 1997 International Conference on*. 1997.

125. Belmonte, F., et al., *Interdisciplinary safety analysis of complex socio-technological systems based on the functional resonance accident model: An application to railway trafficsupervision.* Reliability Engineering & System Safety, 2011. 96(2): p. 237-249.

126. Leveson, N., *Engineering a safer world : systems thinking applied to safety*. Engineering systems. 2011, Cambridge, Mass.: MIT Press. xx, 534 p.

127. Fleming, C., et al., *Safety Assurance in Nextgen.* NASA Technical Report, 2011.

128. Leveson, N.G. *A new approach to hazard analysis for complex systems*. in *International Conference of the System Safety Society*. 2003. Ottawa, Ontario Canada.

129. Ishimatsu, T., et al., *Modeling and Hazard Analysis using STPA*, in *Conference of the International Association for the Advancement of Space Safety* 2010: Huntsville, Alabama.

130. Owens, B.D., et al. *Application of a safety-driven design methodology to an outer planet exploration mission*. IEEE.

131. Ishimatsu, T., et al., *Multiple Controller Contributions to Hazards.* 2011.

132. Nakao, H., et al., *Safety Guided Design of Crew Return Vehicle in Concept Design Phase Using STAMP/STPA.* 2011.

133. Leveson, N., et al., *Risk Analysis of NASA Independent Technical Authority*. 2005.

134. Pereira, S., G. Lee, and J. Howard, *A System-Theoretic Hazard Analysis Methodology for a Non-advocate Safety Assessment of the Ballistic Missile Defense System*, in *AIAA Missile Sciences Conference*2006: Monterey, CA.

135. Dong, A., *Application of CAST and STPA to railroad safety in China*, 2012, Massachusetts Institute of Technology.

136. Antoine, B., *Systems Theoretic Hazard Analysis (STPA) Applied to the Risk Review of Complex Systems: an example from the medical device industry*, in *Engineering Systems Division*2012.

137. Fleming, C.H., et al., *Safety assurance in NextGen and complex transportation systems.* Safety science, 2013. 55(0): p. 173-187.

138. Thomas, J. and N. Leveson, *Performing Hazard Analysis on Complex, Software- and Human-Intensive Systems*, in *International System Safety Conference*2011, System Safety Society: Las Vegas, NV.

139. Thomas, J., B. Antoine, and N. Leveson, *A Systems Approach to Hazard Analysis of Nuclear Reactor Safety Systems.* (in process).

140. Thomas, J., F. Lemos, and N. Leveson, *Evaluating the Safety of Digital Instrumentation and Control Systems in Nuclear Power Plants*, in *NRC Technical Researcy Report*2013.

141. Syson, D.,*The £3bn child of Chernobyl*, in *Daily Mail* 2008.

142. AREVA NP Inc., *U.S. EPR Final Safety Analysis Report*, 2011.

143. Heimdahl, M.P.E. and N.G. Leveson, *Completeness and consistency in hierarchical state-based requirements*. Software Engineering, IEEE Transactions on, 1996. 22(6): p. 363-377.

144. Leveson, N.G., M.P.E. Heimdahl, and J.D. Reese, *Designing specification languages for process control systems: lessons learned and steps to the future*, in *Proceedings of the 7th*

*European software engineering conference held jointly with the 7th ACM SIGSOFT international symposium on Foundations of software engineering*1999, Springer-Verlag: Toulouse, France. p. 127-145.

145. Jaffe, M.S., et al., *Software requirements analysis for real-time process-control systems.* Software Engineering, IEEE Transactions on, 1991. 17(3): p. 241-258.

146. Laracy, J.R., *A systems-theoretic security model for large scale, complex systems applied to the US air transportation system*, in *Engineering Systems Division*2007, Massachusetts Institute of Technology, Engineering Systems Division.

147. Leveson, N.G., *Intent specifications: An approach to building human-centered specifications.* Software Engineering, IEEE Transactions on, 2000. 26(1): p. 15-35.

148. Zowghi, D. and V. Gervasi, *On the interplay between consistency, completeness, and correctness in requirements evolution (vol 45, pg 993, 2003).* Information and Software Technology, 2004. 46(11).