An STPA Primer

Version 1, August 2013

Table of Contents

Introduction

STPA is a new hazard analysis technique based on systems thinking and a new model of accident causation based on systems theory rather than reliability theory. Although STPA has been published and evaluated in many academic papers, it was not possible in those papers to provide a tutorial on STPA. STPA was also described in a book, *Engineering a Safer World* (abbreviated ESW in this primer), by Nancy Leveson. Since then, there have been many requests for a tutorial description of this analysis technique that includes lessons learned from practical applications, updates since the book went to print, and answers to common questions asked in classes and workshops. This STPA primer is our attempt to do so. It contains a detailed tutorial on how to use STPA, answers to frequently asked questions, and lots of real-life examples of its use.

This primer is not stand-alone, however. Users need to read *Engineering a Safer World* first to understand the foundations of STPA, the principles for analyzing organizational, human, and technical components of sociotechnical systems, the effective integration of STPA into system engineering processes, and other practical lessons. Throughout this primer, an attempt is made not to duplicate what is in Leveson's book but to explain it further. References back to the book are made when something has already been thoroughly discussed there.

Because STPA is relatively new, we and others are still learning from our experience using it, extending it, and applying it to other system properties, such as security. Rather than waiting many years to produce this primer, it will instead serve as a "living book" in the sense that it will be periodically updated with new material and better descriptions as more experience is obtained. We will identify when the updates are made and which sections have been significantly changed.

Few of the traditional hazard analysis techniques, which mostly date from more than 50 years ago, have actually been scientifically evaluated. In fact, there is much evidence that these techniques are not cost-effective, but unfortunately there have been few alternatives. Many experimental comparisons between STPA and traditional techniques have been done as well as less formal comparisons on industry projects. A chapter in the primer is devoted to these results so far and more results will be added in the future.

To provide a more active learning environment, we have inserted exercises and questions for the reader throughout. Our solutions to exercises are provided at the end of the book. Some exercises involve applying the ideas to a system with which you are familiar. In those cases, we have tried to provide some ways you can evaluate your own answers in terms of common mistakes and other evaluation methods and questions. Some questions are inserted just to help you think carefully about the material being presented. In an interactive classroom setting, these are the questions we would ask students to ensure they understand what is being presented. The answers are not provided as that would be impractical. We have also scattered answers to frequently asked questions throughout the primer as we often hear the same questions many times or find many people making the same mistake.

We would appreciate feedback on the primer, such as what is clear, what is not so clear, and what needs to be added. We will use the feedback to improve future versions of this document and to include additional "frequently asked questions" in the primer. Also, there are many papers, reports, dissertations, etc. including full examples on the PSAS website: http://psas.scripts.mit.edu/home/

Chapter 1: What is STPA?

Nancy Leveson

(First version: 9/2013
Change history:)

STPA, or Systems-Theoretic Process Analysis, is a new hazard analysis technique with the same goals as any other hazard analysis technique, that is, to identify scenarios leading to identified hazards and thus to losses so they can be eliminated or controlled. STPA, however, has a very different theoretical basis or accident causality model. As stated in the introduction, STPA is based on systems theory while traditional hazard analysis techniques have reliability theory at their foundation. As explained in Engineering a Safer World, STPA results in identifying a larger set of causes, many of them not involving failures or unreliability. While traditional techniques were designed to prevent *component failure accidents* (accidents caused by one or more components that fail), STPA was designed to also address increasingly common *component interaction accidents*, which can result from design flaws or unsafe interactions among non-failing (operational) components. In fact, the causes identified using STPA are a superset of those identified by other techniques. Many of these additional causes are related to new types of technology (such as computers and digital systems) and higher levels of complexity in the systems we are building today compared to those built 50 years ago when most of the traditional techniques were developed. Chapter 4 contains data from real projects and experimental comparisons to substantiate this claim of the greater power of STPA. But the claim can also be substantiated theoretically by understanding more about accident causality models.

What is an Accident Causality Model?[1]

All hazard analysis is based on some conception by the analyst (and built into the analysis technique) of how and why accidents occur. If accidents were totally random events involving complex and random interactions of various events and conditions, then it would not be possible to proactively identify specific sets of scenarios leading to losses before they occur.

In fact, the idea of random causality is the basis of the epidemiological approach to safety, first suggested by Gordon in the 1940s [Gordon 1954, Thygerson 1977]. While epidemiology is usually applied to disease, Gordon suggested that it could also be applied to accidents and to the injuries that resulted. This epidemiological model of accidents and injury assumes that accidents result from random interactions among an agent (physical energy), the environment, and the host (victim). As in classical epidemiology, this assumption leads to a reactive approach to accident analysis. In *descriptive epidemiology*, the incident, prevalence, and mortality rates for accidents in large population groups are collected and general characteristics such as age, sex, and geographical area are identified. *Investigative epidemiology* uses a different approach where the specific causes of injuries and deaths are collected in order to devise feasible countermeasures.

While this after-the-fact epidemiological model of accident causation assumes some common factors in accidents, they can only be determined by statistical evaluation of accident data. On the positive side,

[1] The reader who is not interested in the philosophy behind STPA can skip to the next section on chain-of-event causality models.

because specific relationships between causal factors are not assumed, previously unrecognized relationships can be discovered. In addition, determinant as opposed to chance relationships can be distinguished through statistical techniques.

Question: What are some examples of systems or conditions in which this epidemiological approach might be particularly useful?

The alternative to an assumption of total randomness is to posit that there is a pattern to accidents that can be used proactively to identify potential causes for accidents in specific system designs. Epidemiologists work with large populations and natural (undesigned) systems. Safety engineers are more likely to be involved in human-designed systems where the structure and relationships in the system are known and, in fact, are designed and documented. Proactive approaches to accident prevention can be created that exploit common patterns in accidents by analyzing a specific system structure for patterns that might lead to a loss.

To identify possible patterns in accidents, the definition of a *cause* needs to be considered.

What is a cause?

Philosophers have debated the notion of causality for centuries. John Stuart Mill (1806-1873) defined a cause as a set of *sufficient conditions*. "The cause is the sum total of the conditions, positive and negative, taken together, the whole of the contingencies of every description, which being realized, the consequence invariably follows" [222].

As an example, combustion requires a flammable material, a source of ignition, and oxygen. Each of these conditions is necessary, but only together are they sufficient. The cause, then, is all three conditions, not one of them alone. The distinction between *sufficient* and *necessary* is important [Lewycky 1987]. An event may be caused by five conditions, but conditions 1 and 2 together may be able to produce the effect, while conditions 3, 4, and 5 may also be able to do so. Therefore, there are two sets of causes (sets of conditions sufficient for the event to occur). Both of the causes (called *causal scenarios* in this document) have a set of necessary conditions.

Question: What are some necessary and sufficient conditions for an accident in your industry?

The phrase "necessary and sufficient" implies direct causality and linear relationships. A causes B implies that if A occurs, then B will occur and that B will not occur unless A does. But there are lots of situations where indirect causality is important, that is, where the relationship is neither necessary nor sufficient. These factors may be labeled as *systemic* causal factors.

As an example, consider drunk driving. Drunk driving is said to "cause" accidents, but being drunk while driving a car does not always lead to an accident. And accidents occur without drivers being drunk. The similar indirect relationship holds between smoking and lung cancer. The Tobacco Institute exploited this distinction for years by arguing that not all smokers get lung cancer and non-smokers also get lung cancer so there cannot be a causal relationship. The answer is that there is a relationship, as most people now agree, but it is not a direct one. The factors involved in the indirect relationship may be well understood (as in drunk driving) or they may be less well established or understood (as in smoking and lung cancer). An indirect causal relationship is one in which X exerts a causal impact on Y but only through a third variable Z. In more complex relationships, the nature of the relationship between X and Y may even vary over time, depending on the value of Z.

Exercise: What are some other examples of indirect causation?

Why does my definition of cause matter?

You may be asking, what does all this philosophy have to do with me? I just want to build and operate safer systems. The answer is that the underlying accident causality model or assumptions you are using

will determine the success of your efforts. An accident model underlies all our efforts to prevent accidents and to perform hazard analysis. You may not be aware that you are using an accident causality model, but you are. Whether you consider accidents as an unfortunate but unavoidable result of random events (as in the epidemiological models), as a result of individual component failures, or as a result of dysfunctional interactions and inadequately controlled processes in the system, these assumptions will determine the types of accidents you can analyze and prevent in your system.

Hazard analysis can be described as "investigating an accident before it happens." To do that hypothetical investigation, some assumptions about the cause of accidents are required. An important assumption is whether the accident model includes only direct causality or whether it includes systemic or indirect causality. One example is that the safety of nuclear power plants rests to a large degree on the *safety system*, whose operation is required to be independent of the non-safety systems in the plant. If a hazard analysis technique is used that identifies only direct relationships, then indirect dependencies may not be identified and independence may be assumed to exist when it does not.

> **Question**: Do the systems on which you work contain assumptions about independence of components or functions that might be compromised by indirect causal relationships?

A hazard analysis method based on systemic causality can identify non-direct dependencies and relationships. A systemic cause may be one of a number of multiple causes, may require some special conditions, may be indirect by working through a network of more direct causes, or may require a feedback mechanism [Hall 1997, Korzybski 1933, Lakoff 2012, Senge 1990]. Systemic causality is especially important when studying ecosystems, biological systems, economic systems, and social systems. Our engineered systems used to be simple enough that considering only direct linear causality was adequate. But complexity has grown to the point where systemic causality must be considered for us to adequately engineer for safety. The traditional model of accidents as chains of direct causal events is no longer adequate.

Traditional Chain-of-Failure-Event Causality Models

The traditional assumptions about and pattern used to explain accidents (the accident causation model) underlying almost all of the traditional hazard analysis method is the chain-of-events model. Accidents are seen as being caused by a chain of failure events over time, each event directly leading (being necessary and sufficient) to cause the following event. Eventually a loss occurs. For example, the brakes fail, which leads to the car not stopping in time, which leads to the car hitting the car in front of it. The events considered are almost always hardware failures, human errors, software "failures", or energy-related such as an explosion. In HAZOP, deviations of system parameters are considered instead of or in addition to failures but direct causality is still assumed. As example of a chain of events description of an accident is shown in Figure 2.4 on page 17 of ESW.

Using this model, the reasonable approach to hazard analysis is to create plausible chains of failure events that can lead to the accident being prevented. These plausible chains can be used to create changes to the system design or operations in order to prevent the failures. As the events involve failures, human errors or uncontrolled energy, it makes sense to try to prevent accidents by increasing the reliability of the system components to prevent the failures and stop the chain from occurring, for example, to increase the reliability of the brakes in the car. Note that reliability of the component behavior, i.e., the prevention of failure events, is the foundation of such an accident causation model. Failures are considered to be random, and therefore it is reasonable to assign a probability to such failure events.

In some industries, primarily the process industry, the standard approach to designing for safety is to put *barriers* between the events, especially the energy related events, to prevent the chain from

propagating even though the failure events may occur, for example, using a containment vessel to "contain" the inadvertent release of some chemical or radioactive materials before it affects a victim or a shield to protect someone from the inadvertent release of energy. The events considered then are the failure of the barriers and not necessarily the basic component failure events. The problem becomes one of reducing the probability of the barriers failing, which is again considered to be random.

In other industries, more general types of prevention measures may be used such as sophisticated fault-tolerant and fail-safe design techniques. The events in Figure 2.5 on page 18 of ESW are annotated with a variety of common design techniques, such as building the tank using a non-corrosive material to eliminate the possibility of corrosion.

Formal chain-of-failure-event models (rather than just implicit models) date back to the 1930s and Heinrich's Domino Model (Figure 1.1) [Heinrich 1931], which focused on operator error as the cause of accidents. As greater understanding of additional factors in accidents started to accumulate, people added new factors into the domino (event) chain, particularly in the 1970s, such as lack of control by management, supervisor behavior, management and organizational structure, and substandard practices. Examples are extensions to the Domino Model by Adams (1976) and by Bird and Loftus (1976).

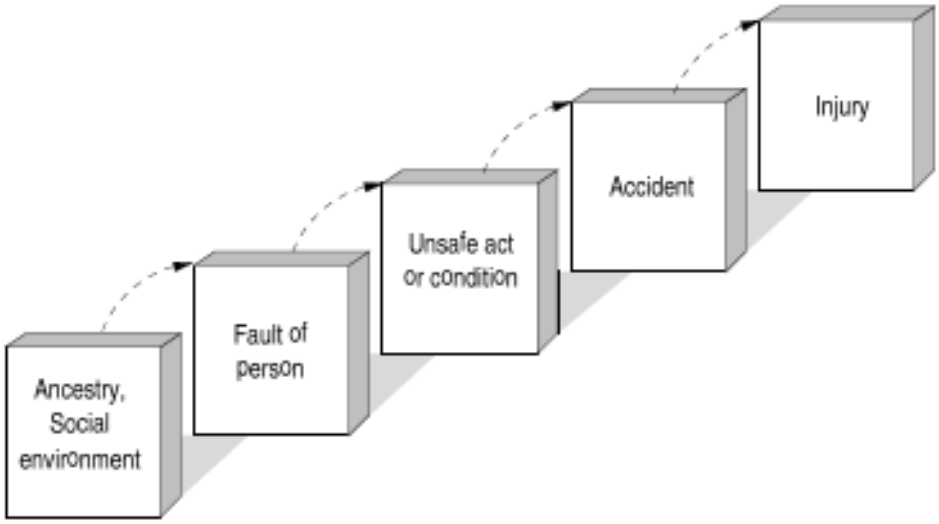


Figure 1.1. The original Domino Model. Notice the focus on direct causality and human error

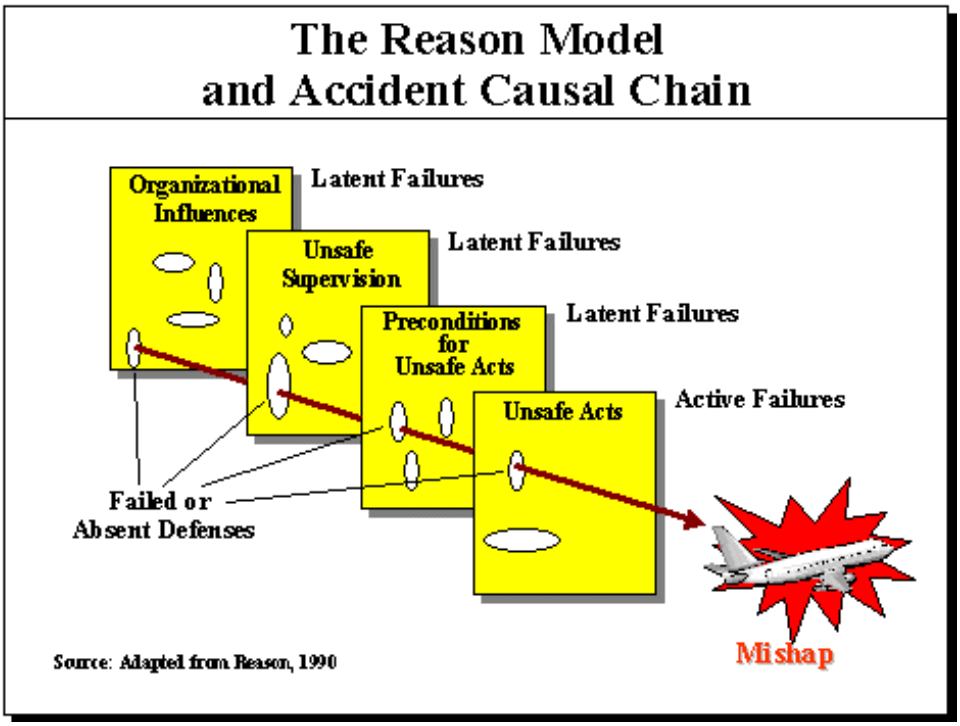


Figure 1.2. One version of the Swiss Cheese Model

In 1990, Reason created the popular Swiss Cheese Model. There are many graphical depictions of his model, with common factor being Swiss cheese slices with accident trajectories shown as an arrow through the holes in the cheese. Different labels may be placed on the slices. A common version is shown in Figure1.2.

The only significant difference between the Swiss Cheese Model and the previous models by Heinrich, Adams, Bird and Loftus, and others was the substitution of Swiss cheese slices for dominoes or other graphics. The holes represent failed or absent barriers or defenses. In an accident scenario, they are considered to line up randomly, which assumes that each slice is independent, which is highly unlikely in any real system. The need to align the holes (failure events) for an accident to occur also implies a precedence requirement for the failure events. The organizational influences must precede (and presumably "cause") unsafe supervision which causes the preconditions for unsafe acts and then finally the unsafe acts and the mishaps. Note the emphasis on human error again, although some versions of the Swiss cheese model do not necessarily use the same labels on the failure events.

The prevailing chain-of-failure-events model provides the basis for almost all of today's hazard analysis techniques (Fault Trees, Event Trees, HAZOP, and FMEA and FMECA) and the probabilistic risk assessment based on them. It also underlies most of our reliability enhancing design techniques, such as redundancy, barriers, safety margins and overdesign, fail-safe design, etc. All of these analysis and design techniques are based on system component failures and thus reliability theory.

Exercise: Where is the chain of events found in a fault tree, an event tree, HAZOP, and FMEA?

This model and the engineering approaches based on it include assumptions that were reasonable at the time when the traditional hazard analysis techniques were created, primarily in the 1960s, although FMEA goes back farther than that. At that time, systems were composed of pure electromechanical components, which could be effectively decoupled and analyzed independently, resulting in relatively simple interactions among the components. System design errors could, for the most part, be identified

and eliminated by testing. What remained after development were primarily random hardware failures. At the same time, operational procedures were simpler and could be completely specified, and operator error mostly involved skipping a step or performing a step incorrectly.

Reliability and safety were, therefore, closely related in these relatively simple designs, and the chain-of-failure-events causality model was an adequate simplification of how accidents occurred.

The situation has now changed. The use of computers and other new technology has allowed increasingly complex designs that no longer can be exhaustively tested. Critical design errors (usually reflected in requirements flaws) are being identified only during operational use. While software design errors may exist that result in the software not implementing the stated requirements, the role of software in accidents and safety-related incidents is much more likely to result from inadequate software requirements. The software can be perfectly reliable (it does the same thing continually given the same inputs) and perfectly implement its requirements, but it may still be unsafe if the behavior specified by the requirements is unsafe.

The problems are similar for human operators. Assumptions about the role of human operators in safety have always been oversimplified. Most human factors experts now accept the fact that behavior is affected by the context in which it occurs and humans do not "fail" in a random fashion [see, for example, Dekker 2006, Flach 1995, Norman 2002, Rasmussen 1997]. The oversimplification has become less applicable to modern systems as operators increasingly assume supervisory roles over automation, which requires cognitively complex decision making where mistakes can no longer be effectively treated as simple random failures. The design of systems today is leading to new types of operator errors, such as mode confusion, that stem from system design and not from random errors on the part of the operator.

The basic problem is complexity. Complexity has increased in current advanced engineering systems to the point where all the potential interactions among system components cannot be anticipated, identified, and guarded against in design and operations. *Component interaction accidents* (as opposed to *component failure accidents*) are occurring where no components have "failed" but a system design error results in accidents caused by previously unidentified, unsafe component interactions and component requirements specification errors. Hazard analysis techniques based on reliability theory and assumptions that accidents are caused by component failures do not apply to component interaction accidents.

There are other limitations of the traditional accident causation models that limit their effectiveness in understanding and preventing accidents. For example, no account is made for common causes of the failures of the barriers or the other types of events in the chains. These "systemic" accident causes can defeat multiple barriers and other design techniques that are assumed to be independent. In addition, no account is taken of Rasmussen's observation that major accidents are not usually the result of simple chains of random events or random failures but instead represent the systematic migration of the system to states of higher risk. At some point in time, an accident becomes inevitable or, as people often observe in hindsight, an accident "waiting to happen." This migration occurs due to competitive or financial pressures that force people to cut corners or to behave in more risky ways [Rasmussen 1997].

As an example, consider the Bhopal accident. None of the safety devices, for example, the vent scrubber, flare tower, water spouts, refrigeration system, alarms, and monitoring instruments worked. At first glance, the failure of all these devices at the same time appears to be an event of extremely small probability or likelihood. But these "failure" events were far from independent. Financial and other pressures led to reduced maintenance of the safety devices, turning off safety devices such as refrigeration to save money, hiring less qualified staff, and taking short cuts to increase productivity. An audit two years before the accident noted many of the factors involved, such as nonoperational safety devices and unsafe practices, but nothing was done to fix them.

This accident was not just a bunch of holes randomly lining up in Swiss cheese slices but the result of a common cause of all the failures of the Swiss cheese slices and systematic degradation of all the protection devices. One can envision the financial pressures at Bhopal as undermining all the barriers over time: think of a mouse eating the cheese slices until the holes are so large that the slices disappear and almost any event will set off a disaster. There was nothing random about the causes of this accident and about the lack of protection provided by all the safety devices. If only the proximate events to the loss are considered, the events can appear random. But if a longer time line and more causal factors (including systemic or indirect ones) are considered, the appearance of randomness disappears. One of the problems is that these models consider the failure events as the cause but do not look at the reasons the failures occurred so only a limited set of causes is considered.[2] They also assume direct causality and therefore indirect and nonlinear relationships among the events are not considered.

Exercise: What are some systemic causes of accidents in systems with which you are familiar?

STPA is based on a different paradigm called systems thinking and systems theory, that is, systemic causality. As such, it includes not only the basic types of accidents that were handled in the past but also new causes of accidents not included in the traditional accident causation models.

What is Systems Theory?

Systems theory is described in Leveson's book (Chapter 3), so only a short summary is included here. Until the 1940s and 1950s, scientists and engineers used analytical reduction to cope with complexity. In analytic reduction, physical complexity is handled by dividing the system into separate physical components, while behavioral complexity is simplified by considering only separate events over time. The assumptions underlying analytical reduction include assuming that the division into parts does not distort the phenomenon and that the interactions among the subsystems and events are simple and direct.

Alternatively, some systems can be conceived as a structureless mass with interchangeable parts. The law of large numbers is used to describe behavior in terms of averages. The assumption here is that the components are sufficiently regular and random in their behavior that they can be studied statistically.

Unfortunately, most of our safety-critical systems today are too complex for complete analysis and too organized for statistics.

Systems theory was developed to deal with these modern systems. It forms the basis for system engineering, where the whole is considered to be more than the sum of the parts and top-down analysis and development is used. Systems theory deals with properties (called emergent properties) that can only be handled adequately holistically, taking into account all the technical and social aspects. These properties arise in the relationships and interactions among system components or behavioral events. That is, systems theory treats systems as a whole and not the components and events separately.

In systems theory, instead of breaking systems into interacting components, systems are viewed (modeled) as a hierarchy of organizational levels. At the lowest level of road traffic, there are the individual vehicles, such as cars and trucks. At the next level there is the design of the roads, which controls the movement of the individual vehicles and their interactions. At a higher level, one can conceive of the entire highway system including the roads but also the rules and policies imposed on the drivers of the vehicles.

[2] Many accident reports start with recounting the chain of events that led to the loss. But the reports usually go on to describe the reasons why the events occurred, although often in a limited way. Traditional hazard analysis methods, however, stop after identifying the chain of events and rarely get to the complex causes behind these events.

The levels of the hierarchy are characterized by emergent properties. These properties are irreducible in terms of not being able to be defined solely in terms of properties of the individual components. The interaction of individual system components results in emergent behavior. Consider the emergent property of highways called gridlock. By looking only at the individual cars, determining whether they will be involved in gridlock is not possible. Only by looking at all the cars together, their relationships and distances from each other, and the characteristics of other parts of the highway system such as the physical structure of the roads, etc., can it be predicted when and how gridlock will occur. Gridlock is an emergent, system property for highway systems.

Exercise: What are some other examples of emergent properties?

Safety is an emergent property. By examining the components of a nuclear power plant, for example, the individual valves and pipes and wires and containment vessels, it is not possible to determine whether the nuclear power plant will be safe. That requires understanding how the individual components are connected and interact, that is, the entire system design. There may possibly be some local safety properties of the individual components, such as sharp edges that could cut anyone who comes in contact with them, but the hazard of "uncontrolled release of radioactive material" cannot be evaluated by looking at the individual components alone without understanding the overall system design (physical and logical connections) and how the components can interact.

In systems theory, each hierarchical level of a system controls the relationships between the components at the next lower level. That is, the levels impose constraints on the degree of freedom of the behavior of the components beneath them. This concept of constraints on behavior plays an important role in STAMP. Safety properties are controlled by imposing constraints on the behavior and interaction of system components. As an example, in an air traffic control system, one safety constraint is that there must always be a minimum distance between airborne aircraft. By definition, then, accidents occur when the safety constraints are not enforced.

Exercise: What are some of the safety constraints in the systems in your industry? How are they enforced or controlled?

The concept of control is important. Each hierarchical level of a system imposes constraints on and controls the behavior of the level beneath it. One can envision a feedback control loop between the components at any level of the system model and the component(s) at the next higher level of abstraction.

By using systems theory, we can get beyond simple direct relationships among components and consider indirect and nonlinear relationships as well as types of control such as feedforward and feedback control. These more complex relationships between components and events cannot be described easily using only boxes with arrows between them, which naturally imply direct causality. That makes it, unfortunately, impossible to graphically describe the cause of an accident using boxes and arrows without losing important information and relationships. STAMP has been criticized because it does not lead to nice graphical models of the causes of an accident, especially one that fits on one page. We agree that simple, graphical models are very powerful, but they lose important information when they show only direct relationships.[3] At best, multiple types of graphical models (such as system dynamic models to show the dynamics and STAMP type models to show the static structure) will be needed along with natural language to adequately describe the causes of an accident.

[3] System dynamic models are one type of graphical model of the dynamics of a system but they also omit important information, such as the static structure of the system. In fact, system dynamics and STAMP have been applied in combination in the past. See, for example, "Demonstration of a New Dynamic Approach to Risk Analysis for NASA's Constellation Program" by Dulac et al.

What is Systems Thinking?

Systems thinking is a term that denotes processes and ways of thinking that follow the principles of systems theory and incorporate systemic causality. Senge (1990) writes:

> [Systems thinking] shifts thinking from blaming a single individual or department, to recognizing that sometimes the problem or fault lies in the entire system and that everybody plays a significant role. Causation becomes multi-causal.
>
> In mastering systems thinking, we give up the assumption that there must be an individual, or individual agent, responsible. The feedback perspective suggests that everyone shares responsibility for problems generated in a system.
>
> With systemic thinking, we recognize that "the cause" frequently lies in the very structure and organization of the system. Such structural awareness enables us to ask, what are the over-arching structures that hold the system together? [Senge 1990, p. 78]

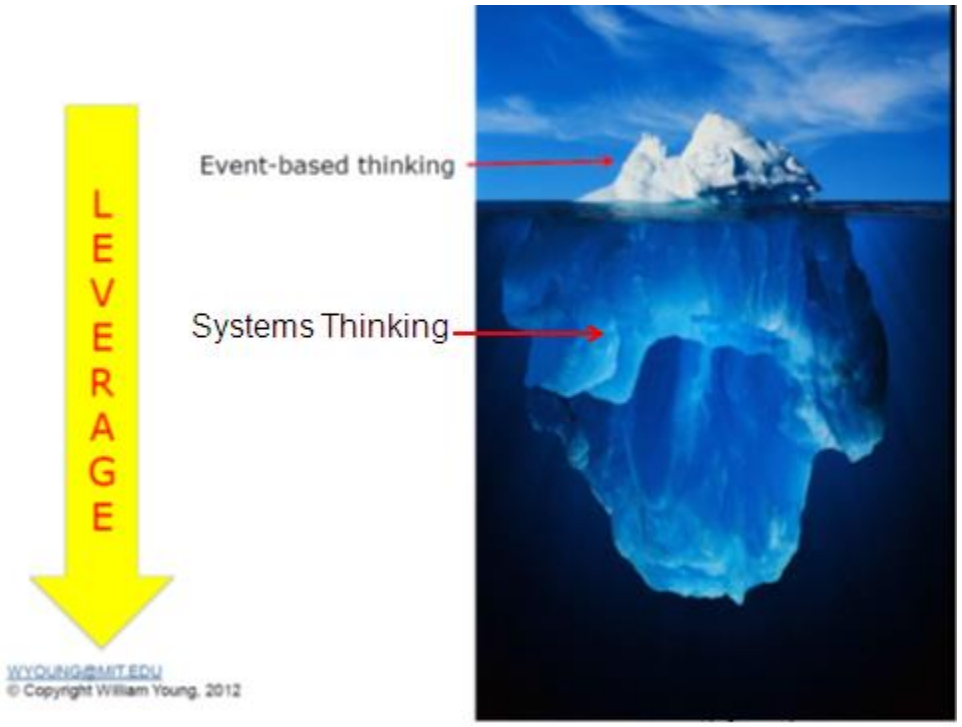


Figure 1.3. Using systems thinking will provide the leverage we need to get beyond simple event-based thinking and reduce accidents in complex systems [Young, 2012]

Engineering a safer world requires not only solving immediate problems but constructing a system that learns and improves over time. “"It is not enough to see a particular structure underlying a particular problem ... This can lead to solving a problem, but it will not change the thinking that produced the problem in the first place." (Senge 1990 p. 95)

By applying systems thinking to safety engineering, we will be able to handle more complexity and more causal factors in safety engineering (Figure 1.3).

STAMP

STAMP (Systems-Theoretic Accident Model and Processes) is a new type of accident model based on systems theory rather than the traditional analytic reduction and reliability theory. In the STAMP model of accident causation, safety is an emergent property that arises when system components interact with each other within a larger environment. There is a set of safety constraints related to the system components—physical, human, and social—that enforces the safety property. Accidents occur when the interactions violate the safety constraints, that is, appropriate constraints are not imposed on the interactions.

The goal of safety engineering, then, is to control the behavior of the components and system as a whole so as to ensure that the safety constraints are enforced. To date, I have not been able to devise a simple graphic, like dominos or Swiss cheese slices, that illustrates STAMP. The problem is that indirect and systemic causality is much harder to depict than simple direct relationships and that STAMP is a total paradigm change from the prevailing accident models today. Figure 1.4 shows my best attempt at creating a graphic to date.

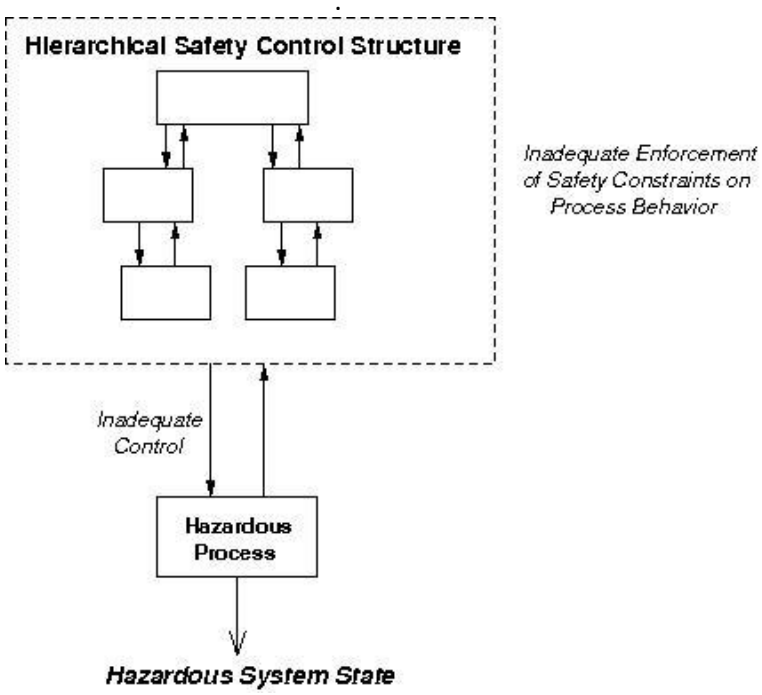


Figure 1.4. Accidents occur when the system gets into a hazardous state, which in turn occurs because of inadequate control in the form of enforcement of the safety constraints on the system behavior

In STAMP, accidents involve a complex, dynamic process. They are not simply chains of component failure events. Safety then can be treated as a dynamic control problem, rather than a component reliability problem. For example, the O-ring in the Challenger Space Shuttle did not control propellant gas release by sealing a gap in the field joint. Yes, the O-ring failed, but the larger problem was not just that failure itself but that the failure led to a violation of a system safety constraint. In other examples, the software did not adequately control the descent speed of the Mars Polar Lander, the Texas City oil refinery design and operations did not adequately control the level of liquid in the ISOM tower, and the problem at the Macondo Well in the Deepwater Horizon fire and oil spill was a similar lack of control

over the pressure in the well. Non-engineered systems can be included, e.g., the financial system did not adequately control the use of financial instruments in our recent financial crisis.

> **Exercise**: For your industry or for a system with which you are familiar, what "process" is being controlled? How is it controlled? What are some typical safety constraints that must be enforced?

The example accidents in the previous paragraph often did include component failures. The point is that they included *more* than just component failures. Unless the accident model defines causality as more than a chain of failure events, other types of factors are missed such as system design errors, software requirements flaws, mistakes in human decision making, migration of the overall system toward states of higher risk, etc. A causality model based on control includes both the failure to control the component failures or their effects and instances where the interactions among components (the overall system design) was the problem and not component failures. STAMP therefore extends the classic model by including it as a subset.

To understand the "why" behind accidents, we need to look beyond just the events to the reasons those events occurred. In essence, STAMP results in a change in emphasis from prevent failures to enforce safety constraints on system behavior (which includes prevent failures but also includes more).

STAMP has three basic concepts: safety constraints, hierarchical safety control structures, and process models. Safety constraints have been discussed. Safety control structures and process models are described briefly here but more details are provided in the next chapter on STPA.

Hierarchical Safety Control Structures

A hierarchical safety control structure is an instance of the more general system theory concept of hierarchical control structure. The goal of the safety control structure (sometimes called the safety management system) is to enforce safety constraints and therefore eliminate or reduce losses.

Figure 1.5 shows an example for a typical regulated industry in the U.S. Only the operations and development control structure are included. Later examples will show other aspects of this structure. Between each level there is a feedback control loop as defined in system theory. Higher level controllers may provide overall safety policy, standards, and procedures, and get feedback about their effects in various types of reports, including incident and accident reports. Lower levels implement those policies and procedures. Feedback provides the ability to learn and to improve the effectiveness of the safety controls.

There are two basic hierarchical control structures in Figure 1.5—one for system development (on the left) and one for system operation (on the right)—with interactions between them. An aircraft manufacturer, for example, might only have system development under its immediate control, but safety involves both development and operational use of the aircraft and neither can be accomplished successfully in isolation: safety must be designed into the aircraft, and safety during operation depends partly on the original design and partly on effective control over operations. Manufacturers must communicate to their customers the assumptions about the operational environment in which the original safety analysis was based, for example, maintenance quality and procedures, as well as information about safe aircraft operating procedures. The operational environment, in turn, provides feedback to the manufacturer about the performance of the system during operations.

Additional control structures might be included that have responsibility over a different aspect of system, such as a control structure to protect the public (control public health) by providing emergency response to violation of safety constraints that can lead to health or environmental consequences.

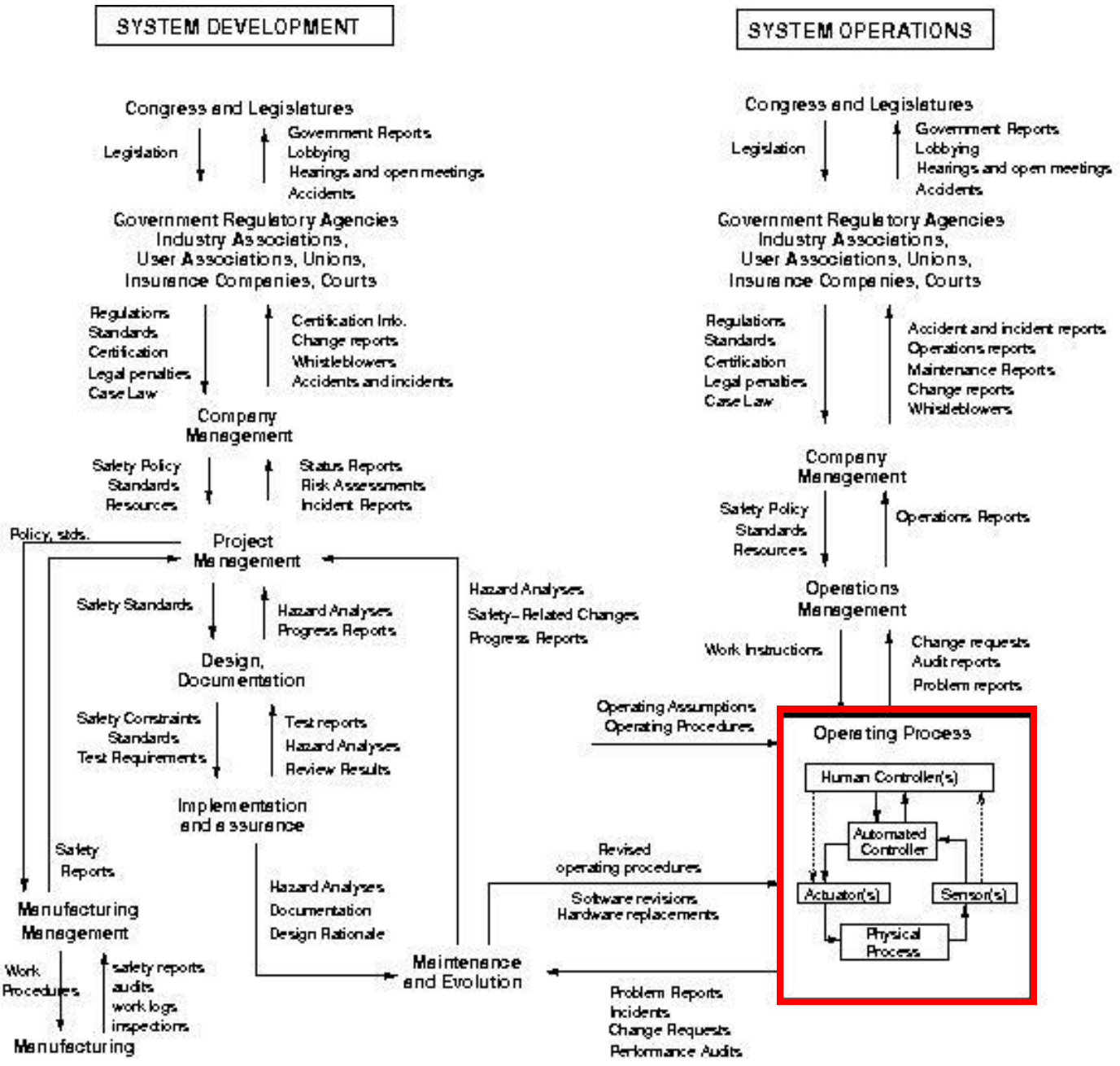


Figure 1.5. An example safety control structure for a regulated industry

Each component in the hierarchical safety control structure has responsibilities for enforcing safety constraints appropriate for that component, and together these responsibilities should result in enforcement of the overall system safety constraint. Part of defining the safety control structure is a specification of the expectations, responsibilities, authority, and accountability with respect to enforcing safety constraints of every component at every level. These responsibilities, authority, etc. taken together must enforce the system safety constraints in the physical design, operations, management, and the social interactions and culture.

Lots of examples of safety control structures can be found in ESW. Figure 1.6 shows a safety control structure (from the Deepwater Horizon accident) that spans companies. One of the problems in that case, as reflected in the finger pointing after the accident, was that the responsibilities were not clearly delineated for all the actors in the system and gaping holes in responsibilities existed.

Control is being used here in a broad sense. Component failures and unsafe interactions may be controlled through design, such as classical redundancy, interlocks, and fail-safe design or more specific types of controls geared to protect against a particular type of behavior. They may also be controlled through process, including developmental, manufacturing, maintenance, and operational processes.

Finally, they may be controlled through various types of social controls. While social controls are usually conceived as being governmental or regulatory, they may also be cultural, insurance, legal, or even individual self-interest. In fact, the most effective way to control behavior is to design (or redesign) a system such that people behave in the desired way because it is in their best interest to do so.

Process models

Control loops exist between every level of the safety control structure, even those at the management and organizational level. Every controller contains an algorithm for deciding what control actions to provide. That algorithm uses a model of the current state of the system it is controlling to help make this decision. Figure 1.7 shows a very simple feedback control loop. The controller is assigned requirements to enforce on the controlled process behavior, which it does by issuing control actions to change the state of the controlled process. For controllers in a safety control structure, the assigned requirements must ensure that the safety constraints are maintained in the controlled process.

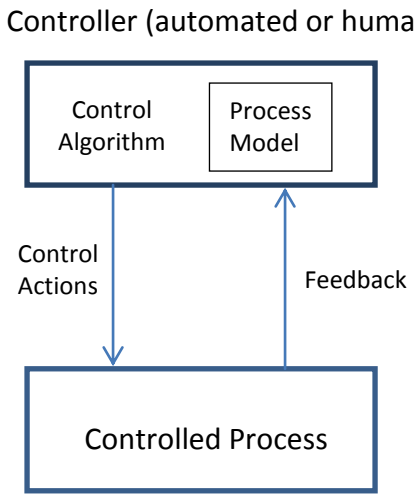


Figure 1.7. Every controller contains a model of the process it is controlling

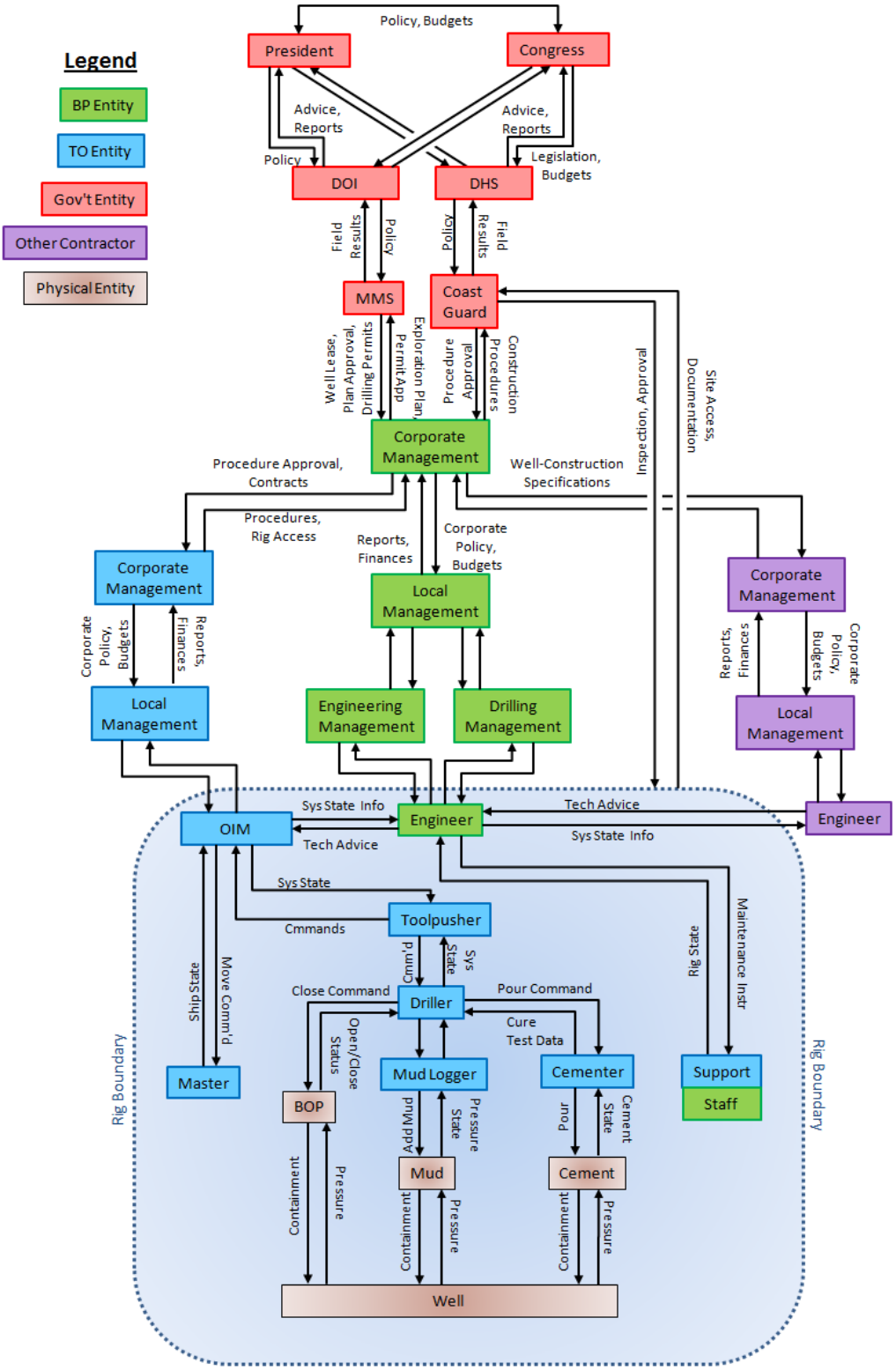


Figure 1.6. The safety control structure over the Macondo well during the Deepwater Horizon accident

In an air traffic control system, for example, the air traffic controller may be assigned responsibility for maintaining safe separation between aircraft. The controller issues advisories to the aircraft to ensure that a loss of minimum separation hazard does not occur.

The *control algorithm* uses information about the process state (contained in the *process model*) to generate those control actions that will cause the process to achieve the requirements (that is, maintain the safety constraints) assigned to that particular controller. In a human controller, the process model is usually called a "mental model." This process model or mental model includes assumptions about how the controlled process operates and the current state of the controlled process.

For example, if a simple thermostat is controlling the temperature in a room, it will determine whether the temperature of the room is at a commanded set point. If not, the controller generates control. One way an accident can occur in such a system is that the controller's process model becomes inconsistent with the real state of the controlled process and the controller provides an unsafe control action to the process. When there are multiple controllers providing control instructions to the same process (including the case where the multiple controllers may be a mixture of humans and computers), accidents can also result when conflicting control actions are provided, perhaps due to inconsistencies between the individual controller's process models. Part of the challenge in designing an effective safety control structure is to provide the feedback and inputs necessary to keep the controllers' models consistent with the actual state of the controlled process and with each other.

There are four general types of unsafe control action:

1. An unsafe control action is provided that creates a hazard (e.g., an air traffic controller issues an advisory that leads to loss of separation that would not otherwise have occurred)
2. A required control action is not provided to avoid a hazard (e.g., the air traffic controller does not issue an advisory required to maintain safe separation)
3. A potentially safe control action is provided too late, too early, or in the wrong order
4. A continuous safe control action is provided too long or is stopped too soon (e.g., the pilot executes a required ascent maneuver but continues it past the assigned flight level)

There is a fifth scenario where a control action required to enforce a safety constraint (avoid a hazard) is provided but not followed. The cause of this fifth scenario will involve inadequate behavior (perhaps a failure or a delay) in a part of the control loop beside the controller, for example, the actuator, the controller process, the sensors, or the communication links.

These five scenarios are a much better model of accident causes related to actions by a human or a computer than is simply a model that says they "failed" with no other information about why. Without understanding the causes of the "failures," options for eliminating or reducing them are limited. STPA uses the four types of unsafe control actions along with the fifth reason for unsafe control to identify potential causes of hazardous behavior, including that involving software or humans. The identified scenarios (hazard causes) can then be used to eliminate the causes from the system or, if that is not possible or practical, to mitigate them. Mitigation might involve changing any part of the control loop (the assigned responsibilities, the design of the controlled process, the control algorithm, the process model, the control actions, designed feedback, a communication link, etc.).

If STPA is used early in the system creation and design process, the results of the analysis can be used to generate the system and subsystem requirements and to create a safer design from the start so that changes do not have to be made late in the design and development process.

Figure 1.8 shows a more detailed (and realistic) model, where a process is being controlled by an automated controller, which, in turn, is being controlled by a human operator (controller).

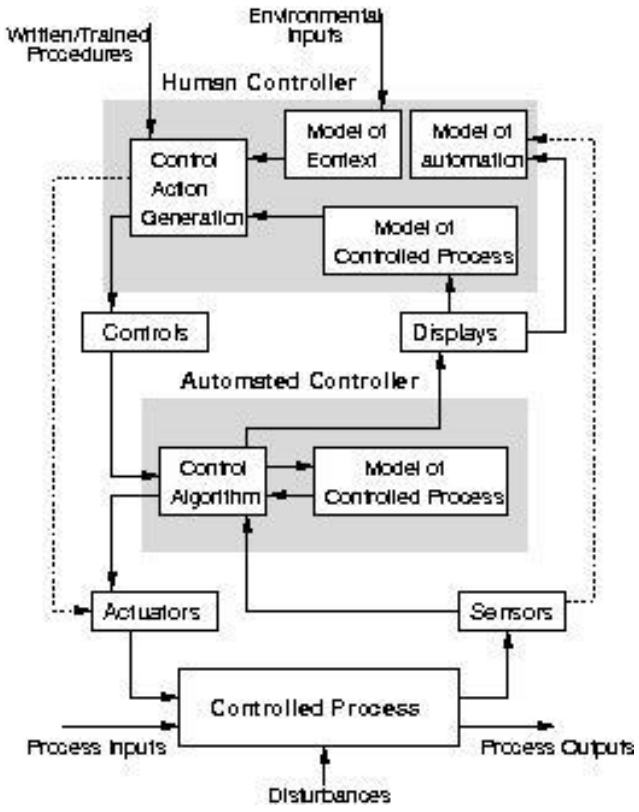


Figure 1.8. A more detailed model of control.

Information about the controlled process state (feedback) is provided by sensors and control actions are implemented on the controlled process by actuators. An automated controller often mediates between human controllers and the controlled process. The human may have direct access to the actuators but more commonly issues instructions to the automated controller through physical or electronic controls. The automated process may also mediate feedback and provide it to the human controller through various types of displays. The dotted lines indicate whether the human controller has direct access to the actuators and sensors or whether all information and control actions must go through an automated device. In a few fully automated systems, there is no human controller directly controlling the physical process although there are humans at higher levels of the control structure.

The control algorithm in the automated controller (which is predesigned and changed infrequently) uses its information about the current state of the controlled process to determine if any control action is needed to be sent to the actuators in order to implement the control requirements, in our case, to enforce the safety constraints. This algorithm is created by a human (not shown in the figure for simplicity) using a process model of what he or she thinks will be the operating states of the controlled process. Unsafe automated control algorithms may result if the designer of that algorithm has an incorrect understanding (process model) of the required behavior of the automated controller.

The process model in the automated controller is updated periodically by feedback from the controlled process that is communicated via sensors and read by the automated controller's control algorithm and used to change the internal process model.

Unlike the automated controller, the human has a control-action generator rather than a fixed control algorithm. Humans may be provided with rules or procedures to follow, but one advantage of having a human in the loop is the flexibility to change procedures or create new ones in a situation that

had not been predicted or to improve the control algorithm without having to go through a long design and implementation process.

One oversimplification in Figure 1.8 is that there is only one controller and one controlled process while there actually may be many of each. Multiple controllers over the same process or processes need some way of coordinating their control actions to avoid a hazardous system state. The control algorithms must be coordinated as well as consistent models of the controlled process state maintained for each controller. Consider a collision between two aircraft that occurred over southern Germany in 2002. Figure 1.9 illustrates the problem that occurred.

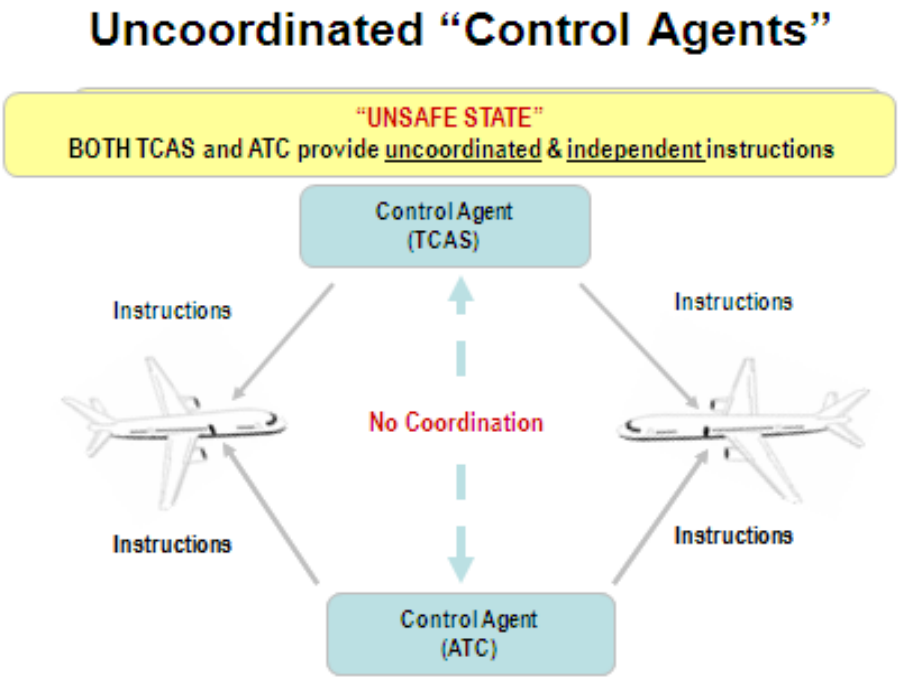


Figure 1.9. The unsafe interactions in the Uberlingen accident.

The ground air traffic controller told one plane to ascend and the other to descend, which would have averted the accident. At the same time, an automated controller (called TCAS or Traffic Alert and Collision Avoidance System) in the two aircraft gave the opposite ascend and descend instructions that the ground controller provided. Again, if both planes had followed the automated controller's instructions, no loss would have occurred. The problem was that one crew followed the air traffic controller's instructions while the other followed the air traffic controller's instructions. As a result, both aircraft descended and collided.

Summary

To summarize, using the STAMP accident causation model, accidents occur when the safety control structure does not enforce the system safety constraints and hazardous states occur due to

1. Unhandled environmental disturbances or conditions
2. Unhandled or uncontrolled component failures
3. Unsafe interactions among components
4. Inadequately coordinated control actions by multiple controllers

The potential for unsafe control may exist in the original design of the safety control structure or the safety control structure and its controls may degrade over time, allowing the system to move to states of increasing risk.

STAMP is only an accident causation model, it is not itself an engineering technique. By using STAMP as a theoretical foundation, however, new and more powerful tools and processes can be constructed. Figure 1.10 shows some of these tools and processes that we have been working on or plan to in the future.

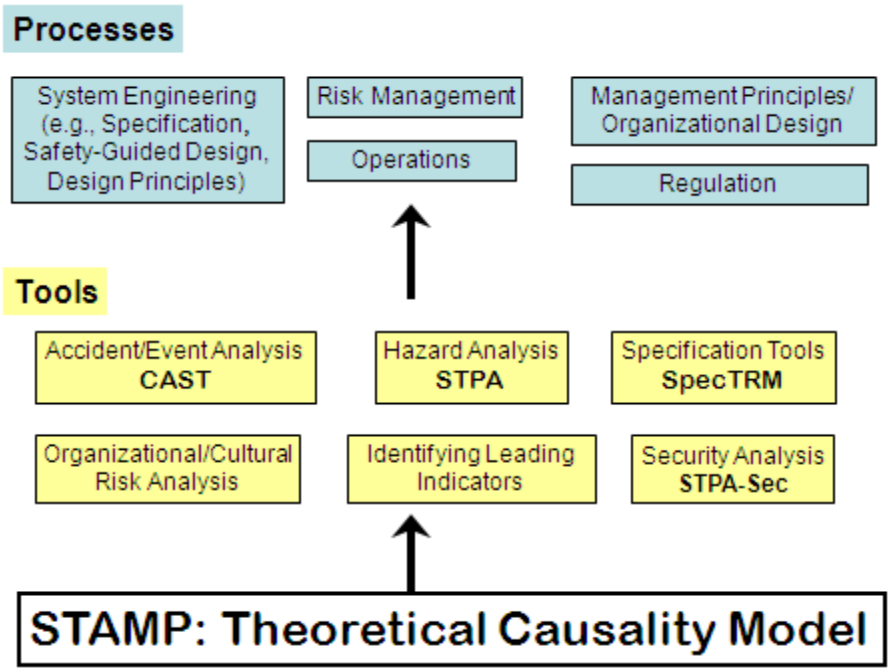


Figure 1.10. Tools and processes can be built upon the STAMP foundation.

Chapter 2: How to Use STPA for Hazard Analysis

Nancy Leveson

(Version 1: September 2013
Change history:)

STPA is a hazard analysis technique that embodies the STAMP accident causality model. As such, it is based on control and system theory rather than the reliability theory underlying most existing hazard analysis techniques. STPA has the same goals as any hazard analysis technique, that is, to accumulate information about how hazards can occur (scenarios). This information can then be used to eliminate, reduce, and control hazards in system design, development, manufacturing, and operations.

STPA does not generate a probability number related to the hazard. The only way to generate such a probability of an accident for complex systems is to omit important causal factors that are not stochastic or for which probabilistic information does not exist (particularly new designs for which historical information is not available). Producing probabilistic analyses that do not accurately reflect the true risk can be dangerously misleading and can lead to complacency and not fixing design flaws that lead to accidents because they are not considered or are unrealistically discounted in importance.

In contrast to the traditional hazard analysis techniques, however, STPA is more powerful in terms of identifying more causal factors and hazardous scenarios, particularly those related to software, system design, and human behavior. This claim can be supported both by theoretical argument and by the experience with its use on a large variety of systems. A few careful comparisons of the results of using the various types of hazard analysis that have been done by us and by others are presented in Chapter 4 of this primer. STPA also provides more support in doing the analysis than most other techniques. Parts can be automated.

Because STPA is a top-down, system engineering approach to system safety, it can be used early in the system development process to generate high-level safety requirements and constraints. These high-level requirements can be refined using STPA to guide the system design process and generate detailed safety requirements on the individual components. In safety-guided design:

- The hazard analysis influences and shapes early design decisions and
- The hazard analysis is iterated and refined as the design evolves.

This safety-guided design process is extremely useful as the cost of rework when design flaws are found late is enormous. When hazard analysis can be done early and in concert with the design decisions, the cost becomes negligible. Note that because STPA can be performed early, it can be used to perform a Preliminary Hazard Analysis. An example is shown later.

STPA can, of course, like the other techniques be used on a completed design or an existing system. As a special case, STPA can be used (like other hazard analysis techniques) in the investigation of accidents in order to generate causal scenarios that can be evaluated for their relevance to the actual events that occurred.

There are some other unique features of STPA. Because it works on the hierarchical safety control structure, it can be used both on technical design and on organizational design. For example, the effect of management decision-making and behavior on accidents can be identified. It also can and has been used on classic project risk analysis and other types of risk analysis.

This chapter first describes the basic STPA process including exercises for the reader and frequently asked questions. Then the use of STPA in special ways (PHA, management and project risk analysis) is described. The chapter concludes with some new extensions that allow incorporating sophisticated human factors concepts into STPA causal scenario generation.

How to do an STPA (The STPA Process)

STPA supports and builds on top-down system engineering. This fact should not be a surprise as systems theory provides a common theoretical foundation for both. The process can be separated into four parts, although the various activities could be intertwined and, in the most effective uses, STPA becomes an iterative process with detail added as the system design evolves :

1. Establish the system engineering foundation for the analysis and for the system development
2. Identify potentially unsafe control actions
3. Use the identified unsafe control actions to create safety requirements and constraints
4. Determine how each potentially hazardous control action could occur.

Establishing the System Engineering Foundation

STPA starts from the basic early system engineering activities associated with safety: defining what accidents or losses will be considered in development, identifying the hazards associated with these accidents, and specifying safety requirements (constraints). After this foundational information is specified, a special STPA process is added: drawing the preliminary (high-level) functional control structure. The actual STPA analysis will use this control structure.

Accidents

Accidents can be defined very narrowly, for example, involving death of humans, or more broadly to include other types of losses. A broad definition will allow the application of safety engineering techniques on a larger variety of problems.

> *An accident is an undesired and unplanned event that results in a loss, including a loss of human life or human injury, property damage, environmental pollution, mission loss, financial loss, etc.*

The term *mishap* has been used, particularly in the defense realm, to reflect this larger definition. In order to avoid a proliferation of terms for no good reason, we use the more common term *accident* and define it to be inclusive. Because what will be considered an accident is the first step in any safety engineering effort, nothing is lost by having only one definition and perhaps it will encourage more broad use of safety engineering techniques.

The determination of what is to be considered as a loss or accident in a particular system has to be made by those assigned such responsibility because it involves the allocation of resources and effort, and these things are never unlimited. For some types of extremely dangerous systems, such as nuclear weapons, the government usually makes this determination. In some industries where safety is critical to the survival of the industry, such as commercial aviation, often the decision is made by national or international associations. Alternatively, the decision may simply be local to a particular company, it may be a requirement imposed by insurance companies, or it may result from liability concerns.

In any case, the definition of what is to be considered an accident or unacceptable loss in a system must be made before any safety efforts begin because it determines the goals and scope of the efforts. Examples of what are commonly considered to be accidents in various types of systems are shown in Table 1.

Table 1: Examples of accidents and hazards.

System	Accident	Hazard Examples
ACC	Two vehicles collide	Inadequate distance between vehicle and one in front or in back
Chemical Plant	People die or are injured due to exposure to chemicals	Chemicals in air or ground after release from plant
Train door controller	Passenger falls out of train	1. Door is open when train starts 2. Door is open while train is moving 3. Door cannot be opened during an emergency 4. Door closing while someone is in the doorway
Unmanned Spacecraft	Loss of mission Pollution of another planet	1. Mission scientific data is not available to researchers at end of mission 2. Planet has biological contamination of Earth origin 3. Other space missions unable to use shared infrastructure to collect, return or use data 4. Toxins, radioactivity, or dangerously high energy levels on Earth or near the ISS after launch of the spacecraft.

Once a decision about the losses to be considered is made, the hazards associated with those losses can be identified.

Hazards

As in traditional System Safety, everything starts from hazards. This term is often used differently in different industries, so a definition of what is meant here is necessary. Where the concept of a hazard is not used or is simply equated to a "failure," then safety is not being adequately addressed and reliability is being substituted for safety. Neither of these two different system qualities implies the other so the substitution means that safety is not being addressed, as discussed in great detail in *Engineering a Safer World*.

Additional definitions of the term "hazard" exist. Many are vague and not workable, such as "A hazard is a condition that is a prerequisite to [or could lead to] an accident or incident" [FAA ATO SMS, MIL-STD-882C]. The drawback of that definition is that there are a very large if not infinite number of conditions that precede an accident. Aircraft being in controlled airspace is prerequisite to an accident or incident, but we cannot eliminate that condition from an air traffic control system, i.e., not allow any

planes in the airspace. Similarly, a prerequisite for (or a condition that could lead to) a collision between two automobiles is that more than one automobile is on the highway at the same time.

The definition used in STPA restricts hazards to be conditions or states that nobody ever wants to occur, such as a violation of minimum separate standards between aircraft in controlled airspace or inadequate braking distance between automobiles in a cruise control system. These conditions, once they are identified, can be eliminated or controlled in the system design and operations. All prerequisites to an accident cannot be considered (and do not need to be) as they include almost all conditions that occur during normal operations.

In practice, we suspect that the actual hazards identified in any hazard identification process will be satisfy the more limited definition used for STPA. Otherwise, generating a list of hazards would be impossible.

STPA uses the following definition in order to be clearer about what to include in the list of hazards:

> *Hazard: A system state or set of conditions that together with a worst-case set of environmental conditions, will lead to an accident (loss).*

There are two important aspects of this definition. The first is that a hazard should be within the system boundaries over which we have control. For example, a hazard for an aircraft is not a mountain or weather because the designer of the aircraft or the air traffic control system has no control over the weather or the placement of a mountain. Instead, the hazard may be the aircraft getting too close to the mountain or the aircraft being in an area of bad weather. Both of these definitions provide potential ways to avoid the hazard when we are designing our system. Another way of saying this is that the hazard must be in the design space of those engineering the system or in the operational space of those operating it.

The second part of the definition is that there must be some worst-case set of conditions in the environment that will lead to a loss. If there is no set of worst case conditions outside or inside the system boundary that will combine with the hazard to lead to a loss, then there is no need to consider it in a hazard analysis. Even if two aircraft violate minimum separation, the pilots may see each other and avoid a collision, but there are also worst case conditions under which the accident may not be avoided such as low visibility, lack of attention by the flight crew, and angles where the other aircraft cannot be seen. Therefore, it is a hazard.

> **Exercise**: For your industry or for a system with which you are familiar, what "process" is being controlled? How is it controlled? What are some typical hazards that must be handled? What are some safety constraints that must be enforced?

If a hazard in your list includes the word "failure," it is almost certainly wrong. First a "failure" is an event; it is not a system state. A failure could lead to a hazardous system state, but it is a possible cause of a hazardous state, not the state or hazard itself. Separating hazards and their causes is important for reasons to be explained later.

In general, words like "failure" and "error" should be avoided, even as causes, because they provide very little information and are often used to avoid having to provide more information or to think hard about the cause. That information is needed to eliminate or control the problem effectively. "Operator error" is much less informative, for example, than "operator thinks the alarm is spurious and ignores it." Even "brakes fail" is less useful than "brakes do not operate when the brake pedal is depressed" or "brakes do not stop the car within the standard braking distance" which could both be effects of failures or of design flaws. We tend to use the words "fail," "failure," and "error" too often as a way of not providing any real information about what happened.

A system hazard is a system-level state. Any mention of subsystems (such as subsystem failure) is not a system state, although it can lead to one (be a cause). At the time the hazards are identified, no

detailed design, including the components exists. For example, “brakes or throttle operate spuriously” is not a hazard (using our definition) for a cruise control system although uncommanded acceleration or deceleration is. Spurious component operation would then be identified in later steps as one potential causes of that hazard. The STPA process includes refining hazards by breaking them down and assigning them to the various components.

Even if an analysis is being done after the design or even the system already exists, starting by specifying the system-level hazards (and not the hazardous behavior associated with the subsystems) is important. Changes may need to be made in the design, including changing the components or their responsibilities. If the analysis starts from hazards specified in terms of the components of the system, changes involving other components are much less likely to be considered by those trying to eliminate the problem.

Exercise: Identify the system-level hazard(s) for the batch reactor shown on Page 9 of ESW.

In your solution to the exercise, was the software controller or the reflux condenser mentioned? They should not have been as they are system components. In fact, the control over the valves might be assigned to the human controllers or another way of maintaining a safe reaction temperature could be used besides a reflux condenser. The analysis will get to that level of detail later. For example, if the high level hazard mentions the reflux condenser, the solution space may be limited to fixing the design of the reflux condenser rather than considering other more effective ways of controlling the temperature in the reactor vessel that do not involve a reflux condenser at all.

The hazard should be a statement about the system as a whole, so the subject should be the batch reactor. The condition or state in the hazard should be a property of the system, so it could refer to the temperature being too hot, the pressure being too high, the reaction being uncontrolled, or the release of toxic chemicals. In fact, systems often have more than one hazard so you may have defined other system states that are dangerous and must be avoided. Also be sure the hazards are things that are controlled by the design of the batch reactor. If the hazards refer to things like the wind speed and direction, the nearby population, or other environmental factors then it is probably not a hazard for the batch reactor system.

The list of hazards should be very small, less than 10 and certainly less than 20. If you have more than that in your list, then you are starting at too low a level of abstraction. In system engineering, the goal is to start at a high-level of abstraction and then refine each level into a more detailed level. That way you are less likely to miss something or to have gaps or redundancies in your list.

FAQ: What is a small number of hazards? Why do I need to limit the number I consider?

Why not start with a long list of hazards, that is, by just generating every potentially unsafe system state and cause one can think of? This is often the strategy used, but it is then difficult (and perhaps impossible) to determine whether anything has been missed—the list is too long and at too many different levels of abstraction. One of the most powerful ways human minds deal with complexity is by using hierarchical abstraction and refinement. By starting at a high level of abstraction with a small list and then refining that list with a more detailed list at each step (working top down), one can be more confident about completeness because each of the longer lists of causes (refined hazards or causes) can be traced to one or more of the small starting list (and vice versa).

With traceability, it is also easier for human reviewers to find any incompleteness. We say "more confident" because such a list can never be proven to be complete—there is no formal (mathematical) model of the entire system and how it will operate. Human participation in the

analysis and human review of the results will always be required and, therefore, incompleteness will always be possible. But structuring the process in a way that optimizes human processing and review will reduce any potential incompleteness.

System Safety Constraints/Requirements

Once the high-level system hazards are identified, they can be translated into safety requirements or constraints. This process is very simple but important because it translates the hazards into the requirements and constraints engineers and system designers need in their standard engineering processes.

Examples are shown in Table 2.

Table 2: Examples of hazards and their related safety constraints.

Hazard	**Safety Constraint (Requirement)**
Inadequate distance between vehicle and one in front or in back	Vehicles must never violate minimum separation requirements
Chemicals in air after release from plant	Chemicals must never be released inadvertently from plant
Door is open when train starts	Train must never start while door is open
Door is open while train moving	Train must never open while train is moving
TCAS causes or contributes to a near miss collision (NMAC)	TCAS must provide effective warnings and appropriate collision avoidance guidance about potentially dangerous threats and do so while there is time to avoid the threat

FAQ: What is the difference between a requirement and constraint and why do you seem to use the terms interchangeably?

There are actually several reasons for using both terms. One simple reason is that *constraint* is the term used in systems theory and STAMP, while *requirement* is more commonly used in engineering so the tie to safety of those things labeled as constraints is explicit.

One often useful distinction is to use requirement to mean the behavior required to satisfy the system's mission or goals while constraints describe limitations on how the mission goals can be achieved. Safety can be involved in both when part of the goal or mission of the system is to maintain safety, such as air traffic control. In other systems, the mission goals and safety constraints do not overlap. In a chemical plant, for example, the mission and mission requirements involve producing chemicals while safety constraints limit the way that the chemicals can be produced. Conflicts between goals and constraints can more easily be identified and resolved if they are distinguished.

Another factor is that some government agencies do not allow negative requirements (because they are not testable). At the same time, some safety constraints cannot effectively be changed into a positive "shall" statement. For example, in TCAS, one of the constraints is that TCAS does not interfere with the ground air traffic control system. There is no easy way to state

that constraint as a positive requirement that does not include the word "not." (Try it.) Auditors for system requirements in these cases do not seem to be bothered by the fact that there are things called "System Constraints" that contain "*must not*" as long as the *shall statements* listed as "Requirements" do not contain that forbidden word. Because negative requirements, or "must not" statements, cannot be verified using testing (which is the reason for forbidding them), other types of verification techniques (such as formal analysis) must be used. Making this required difference in the validation process explicit by using a different label is also sometimes helpful.

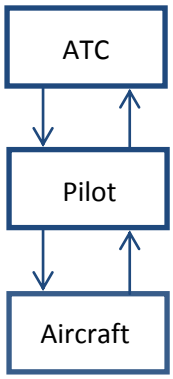
Functional Control Structure

The effort described so far—identifying accidents, hazards, and high-level safety requirements— is common to all safety engineering efforts (or should be) no matter what type of accident causation model or hazard analysis technique is used. STPA unique efforts start at this point. Generating the safety control structure is not part of STPA; it is a system documentation effort needed to perform STPA. While many of the aspects involved in creating the functional control structure will involve standard system engineering activities, such as allocating system requirements to the system components, the use of a functional control diagram to document these decisions is not standard.

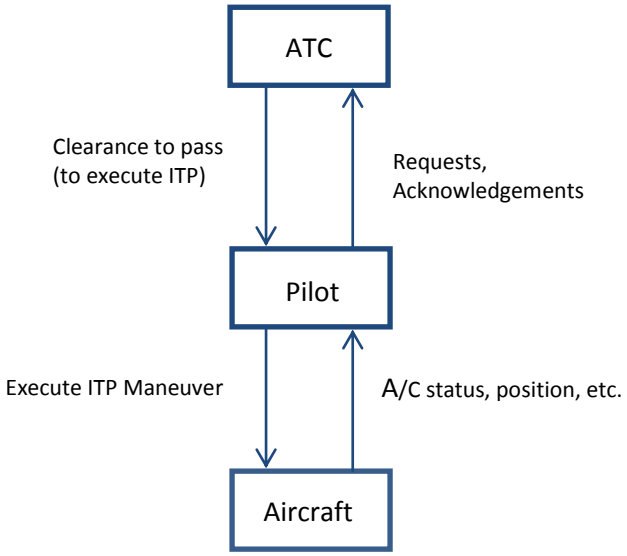
Many people have found that the safety control structure provides excellent documentation and a nice graphical depiction of the functional design of the system. Most complex systems have detailed physical design descriptions and documentation but information about the functional behavior of the system is at best scattered throughout the documentation and sometimes is difficult to understand. The functional control model provides a concise, graphical specification of the functional design.

We have found it easiest for people to understand and to produce functional control diagrams by starting with a very simple, high-level model and then adding detail (refining the model) in steps. The first step may only contain a controller and a controlled process or perhaps a couple of levels of controller (human and automated). For example, a new procedure for allowing one aircraft to pass another over the Atlantic has been designed called In-Trail Procedure (ITP). The pilot first checks that the criteria for performing an ITP (passing) maneuver are satisfied and then asks air traffic control (ATC) for permission to execute the maneuver. The first step in designing or drawing the control structure for ITP might be to identify the main components:

Next, determine who controls who or what.



Once the basic structure has been identified, detail can be added such as the control actions and feedback:



Pilot responsibilities might include:

- Assess whether ITP appropriate
- Check if ITP criteria are met
- Request ITP
- Receive ITP approval
- Recheck criteria
- Execute flight level change
- Confirm new flight level to ATC

The pilot's process model might need to contain:

- Own ship climb and descend capability
- ADS-B data for nearby aircraft (velocity, position, orientation)
- ITP criteria (speed, distance, relative attitude, similar track, data quality)
- State of ITP request/approval
- etc.

Figure 2.2 shows a more detailed control structure for the ITP procedure.

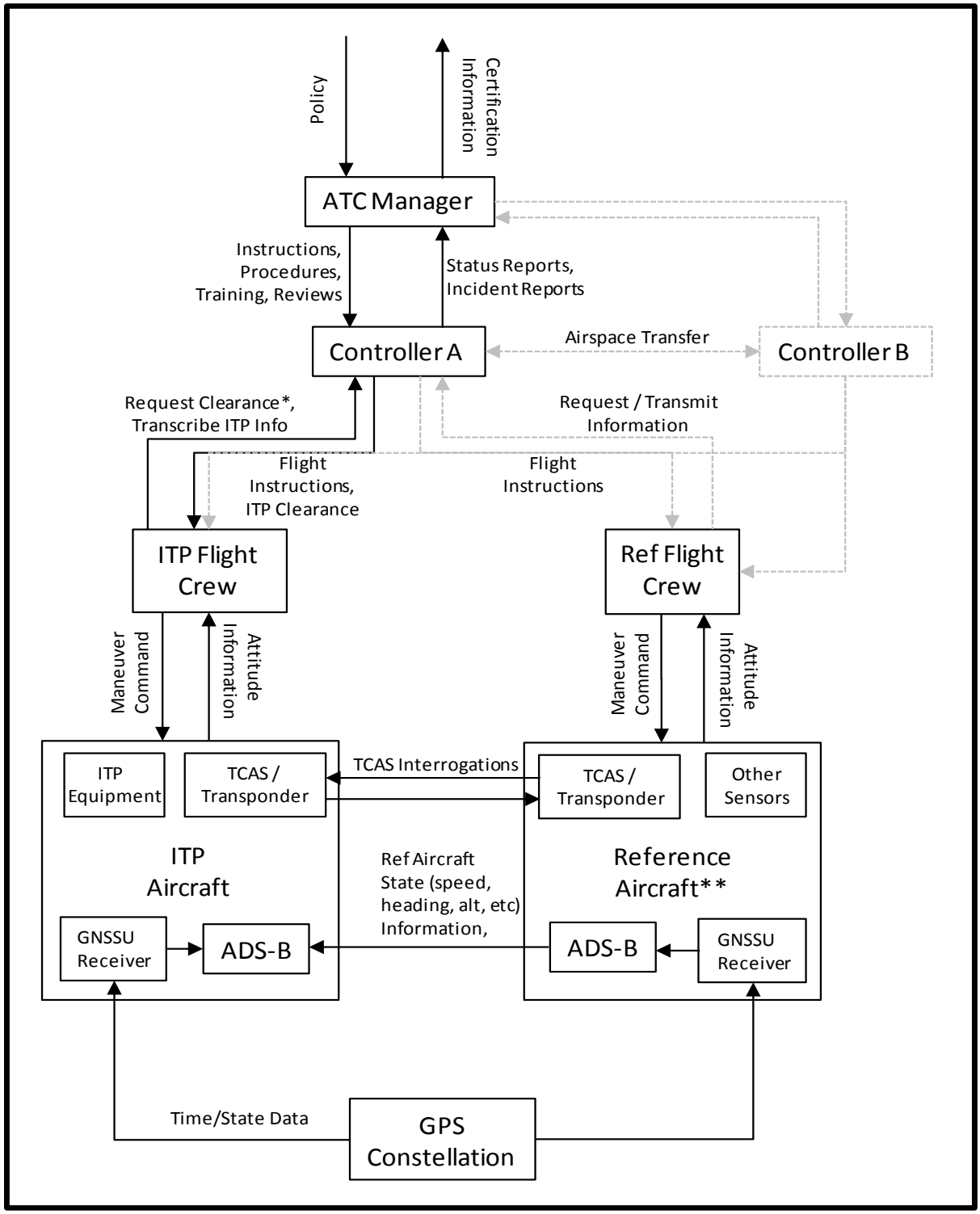


Figure 2.2. ITP Control Structure

For systems where the entire control structure cannot fit on one page, breaking it up into chunks with a high-level model of how the chunks interrelate is very helpful. Figure X is the high-level control structure for the docking of a Japanese spacecraft called the H-II Transfer Vehicle or HTV for short with the International Space Station (ISS).

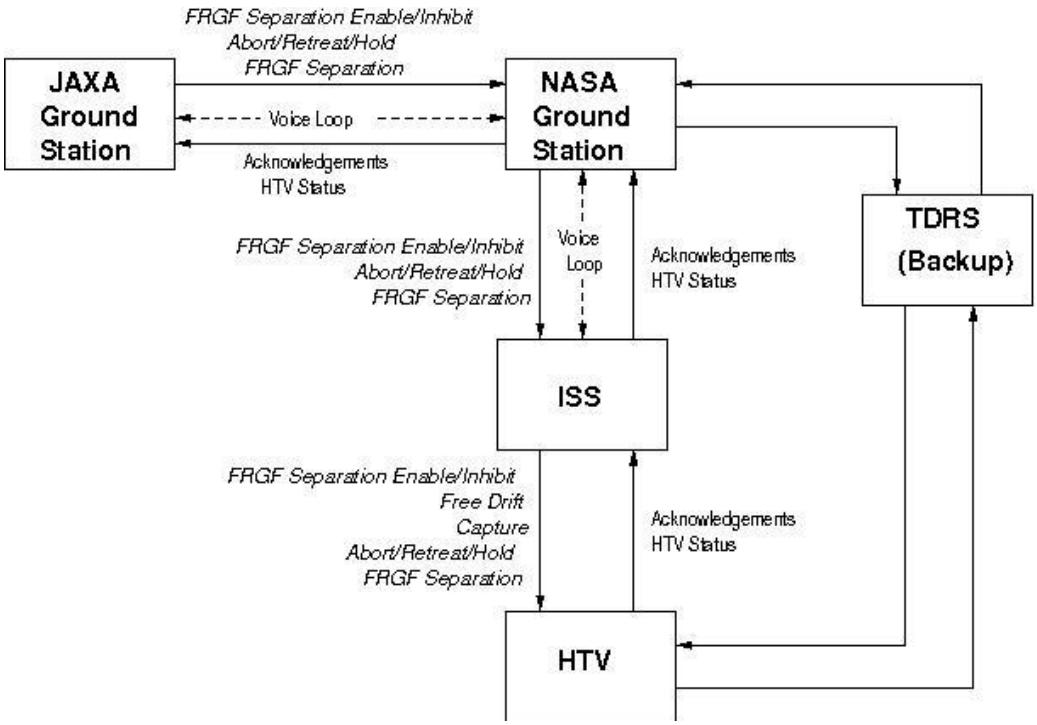


Figure2.3. The high-level control structure for the docking operation of the HTV with the International Space Station ISS).

Figure 2.4 shows a picture of this operation.

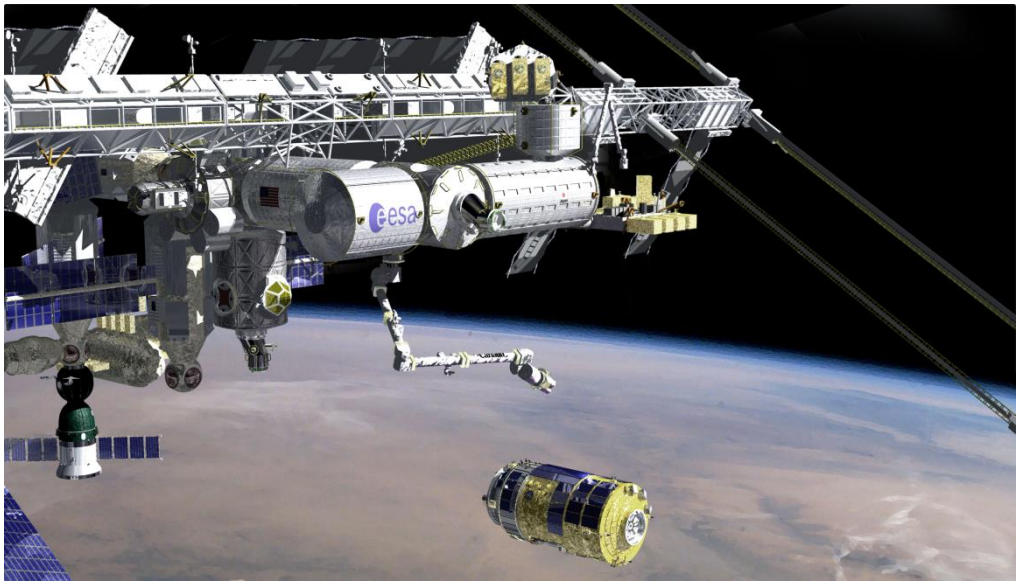


Figure 2.4. A picture of the HTV docking with the ISS.

Five components are shown in Figure 2.3. The ISS, the HTV, the NASA ground station at the Johnson Space Center in Houston, the JAXA ground station at Tsukuba Space Center and TDRS (Tracking and Data

Relay Satellite). The TDRS serves as a backup communications system. The possible control actions and feedback are shown on the diagram.

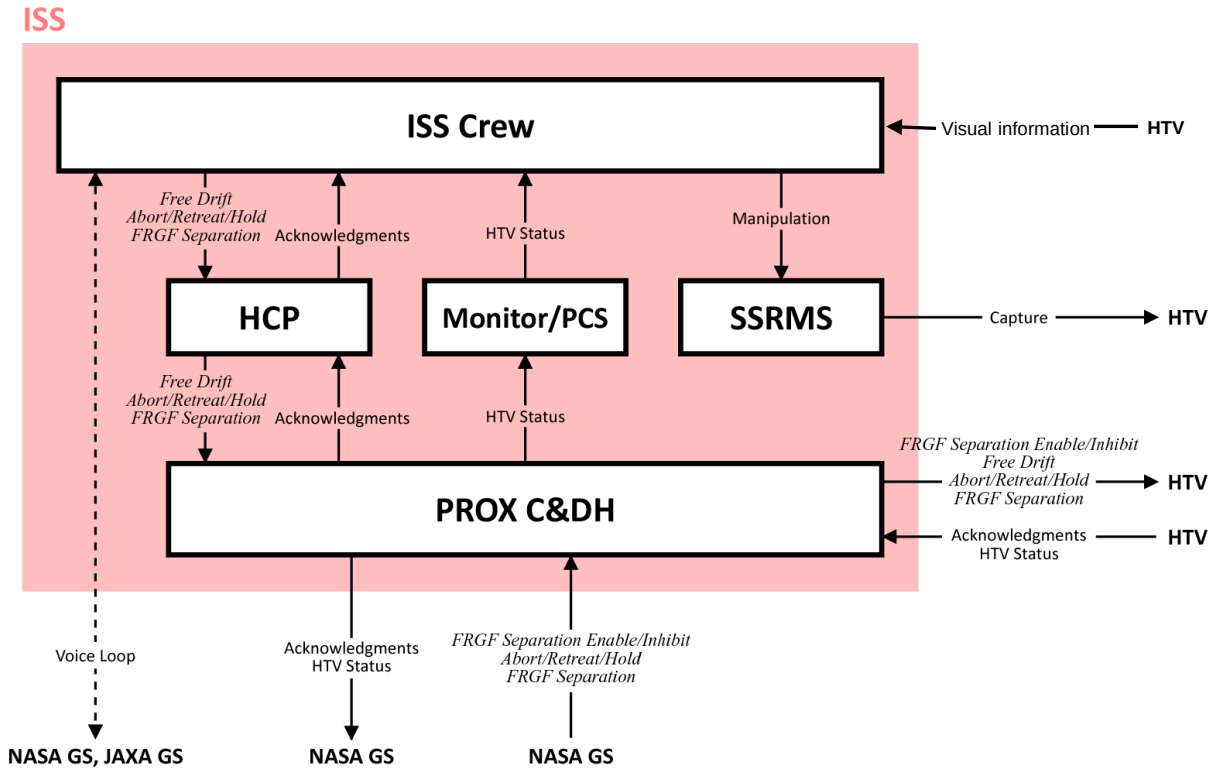


Figure 2.5. A more detailed view of the ISS control structure for HTV docking.

Each of the components in Figure 2.3 can be refined further. For example, Figure 2.5 shows the control structure for the ISS when the HTV is captured by the robot arm (SSRMS). The ISS crew uses the HCP (Hardware Control Panel) to provide instructions to computer software (PROX C&DH) and receive feedback from the computer via a computer display.

When we were analyzing the safety of an important component of a nuclear power plant design for an NRC research project, we were only going to perform STPA on the safety system, but we needed to understand the context within which the safety system existed in order to find indirect and previously unidentified interactions between the non-safety and safety systems. So we started with a high-level control structure of the entire plant.

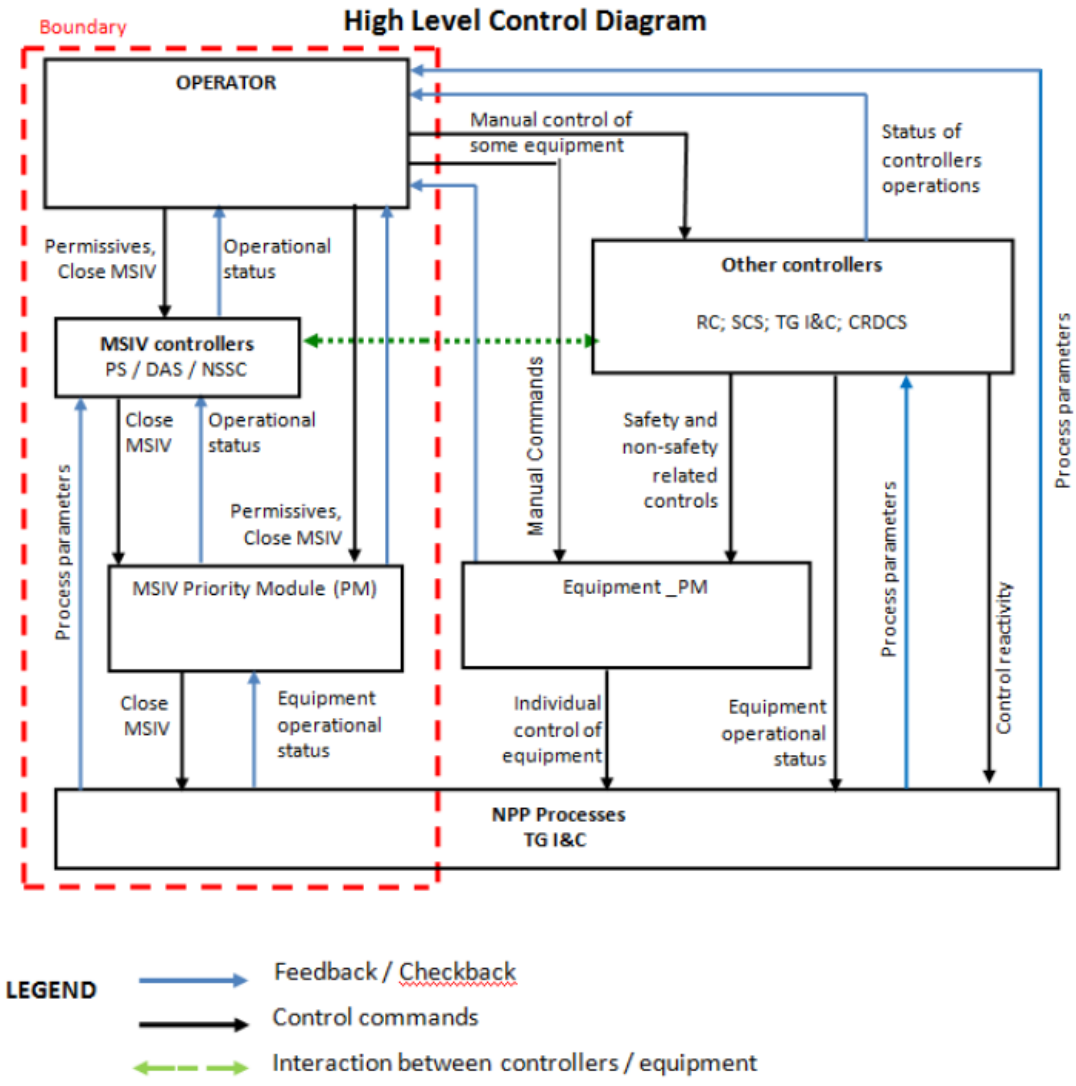


Figure 2.6 shows a refined view of the part outlined by a dotted red line above.

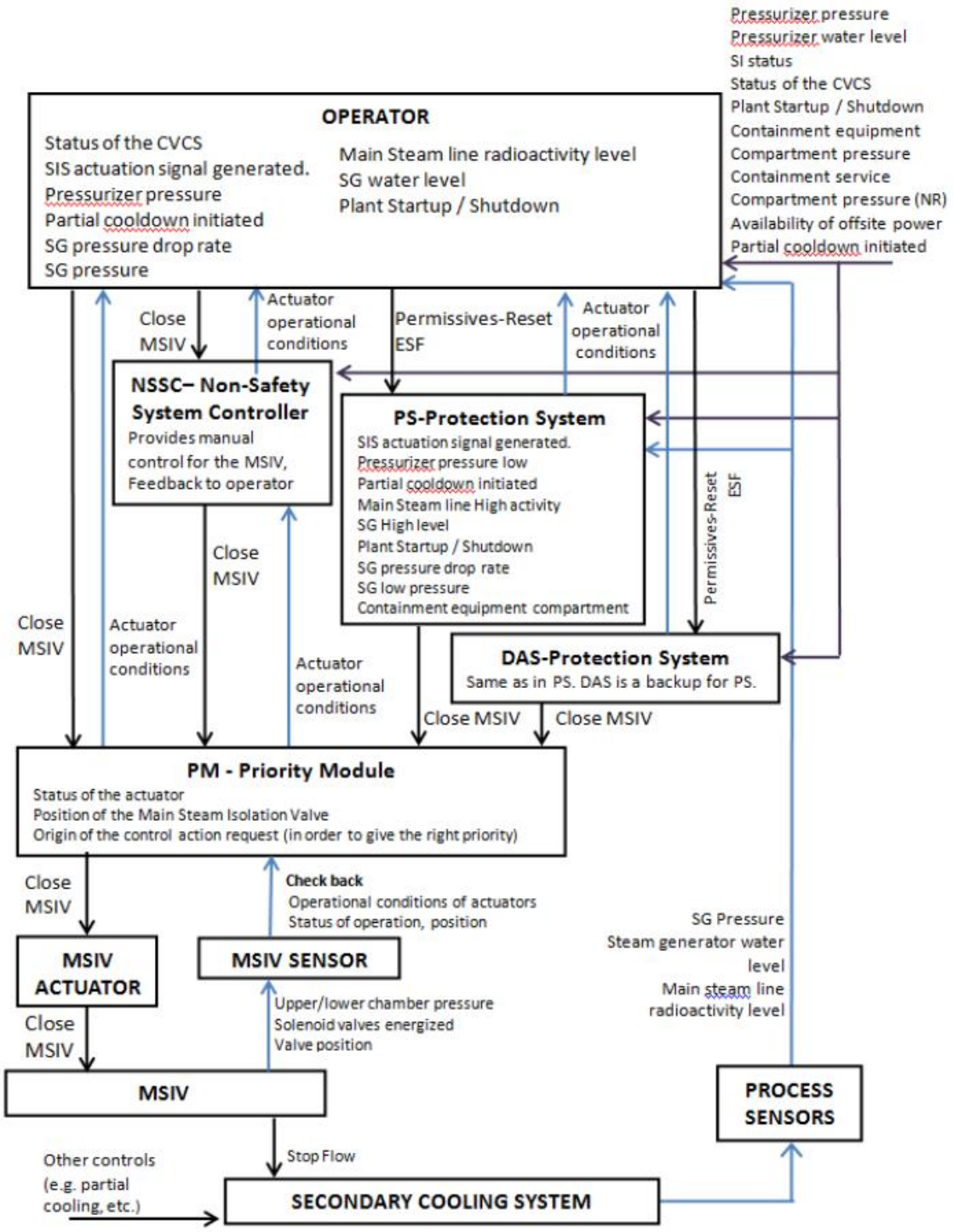


Figure 2.6. More detailed view of the safety system.

Another example involves radiation therapy and a machine called the Gantry 2 that uses protons to treat patients. Figure 2.7 shows a picture of the treatment facility that a patient sees. Figure 2.8 shows the high-level control structure for the system as a whole, including the people involved in creating the treatment plan for the patient and those involved in delivering the treatment.

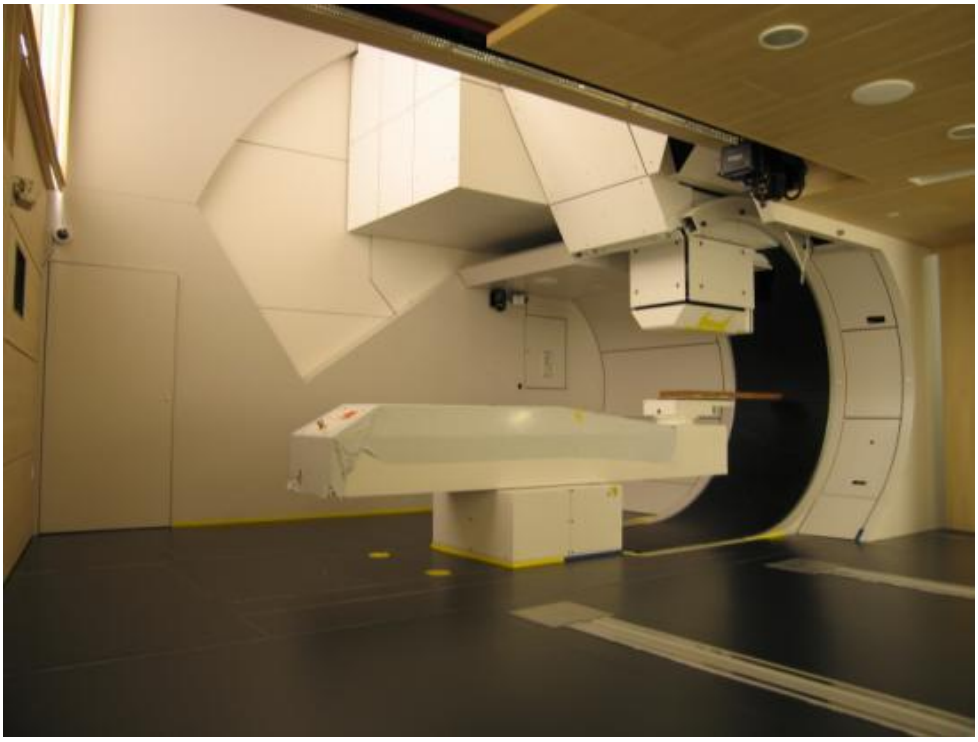


Figure 2.7. The Gantry 2 Treatment Facility

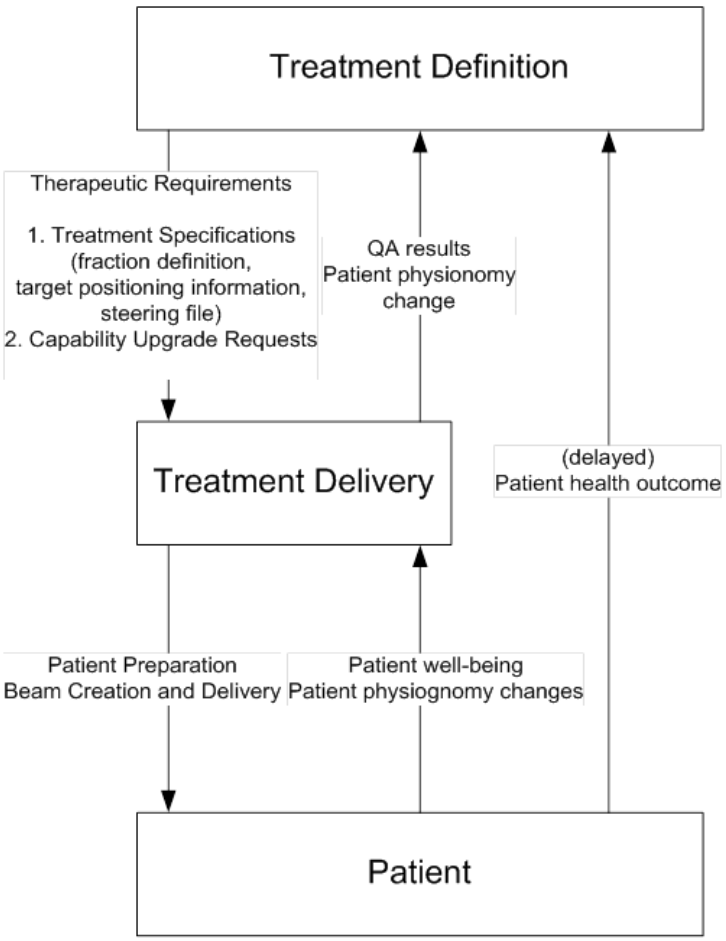


Figure 2.8 : High-Level Control Structure

Each component here can be decomposed. Having multiple levels of abstraction not only assists in creating the control structure but also assists others in understanding it.

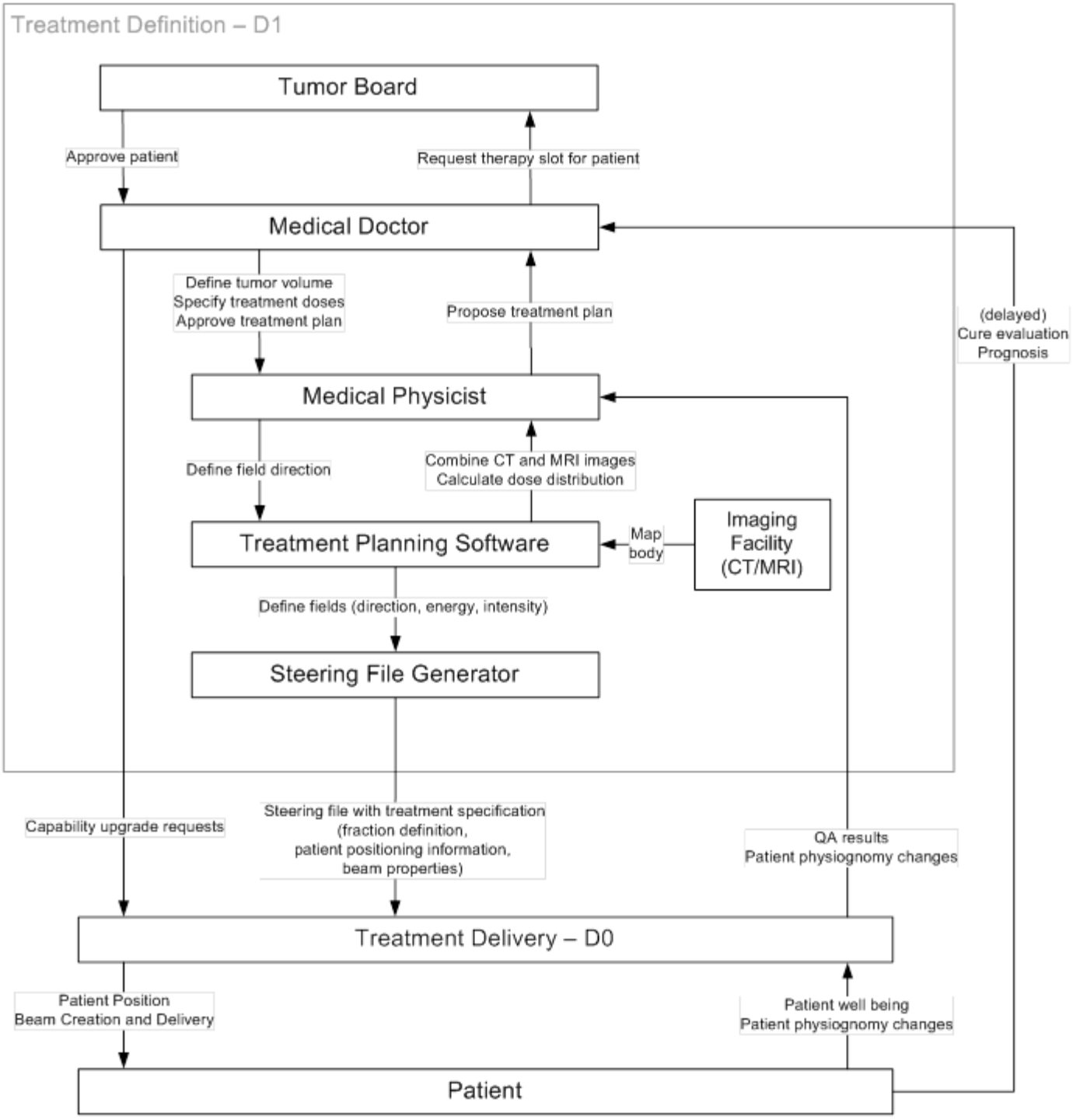


Figure 2.9. Zooming into Treatment Definition

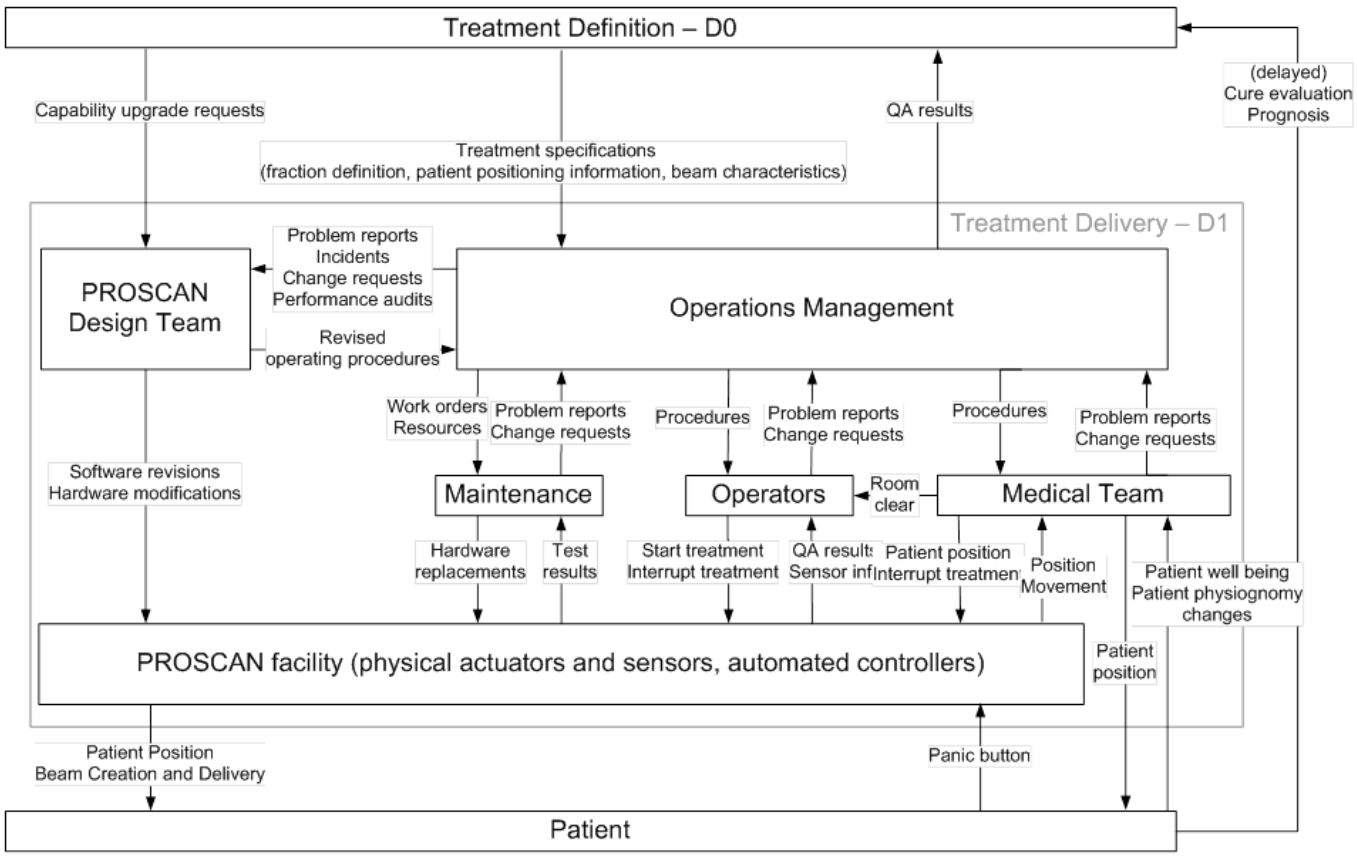


Figure 2.10. Zooming into treatment delivery.

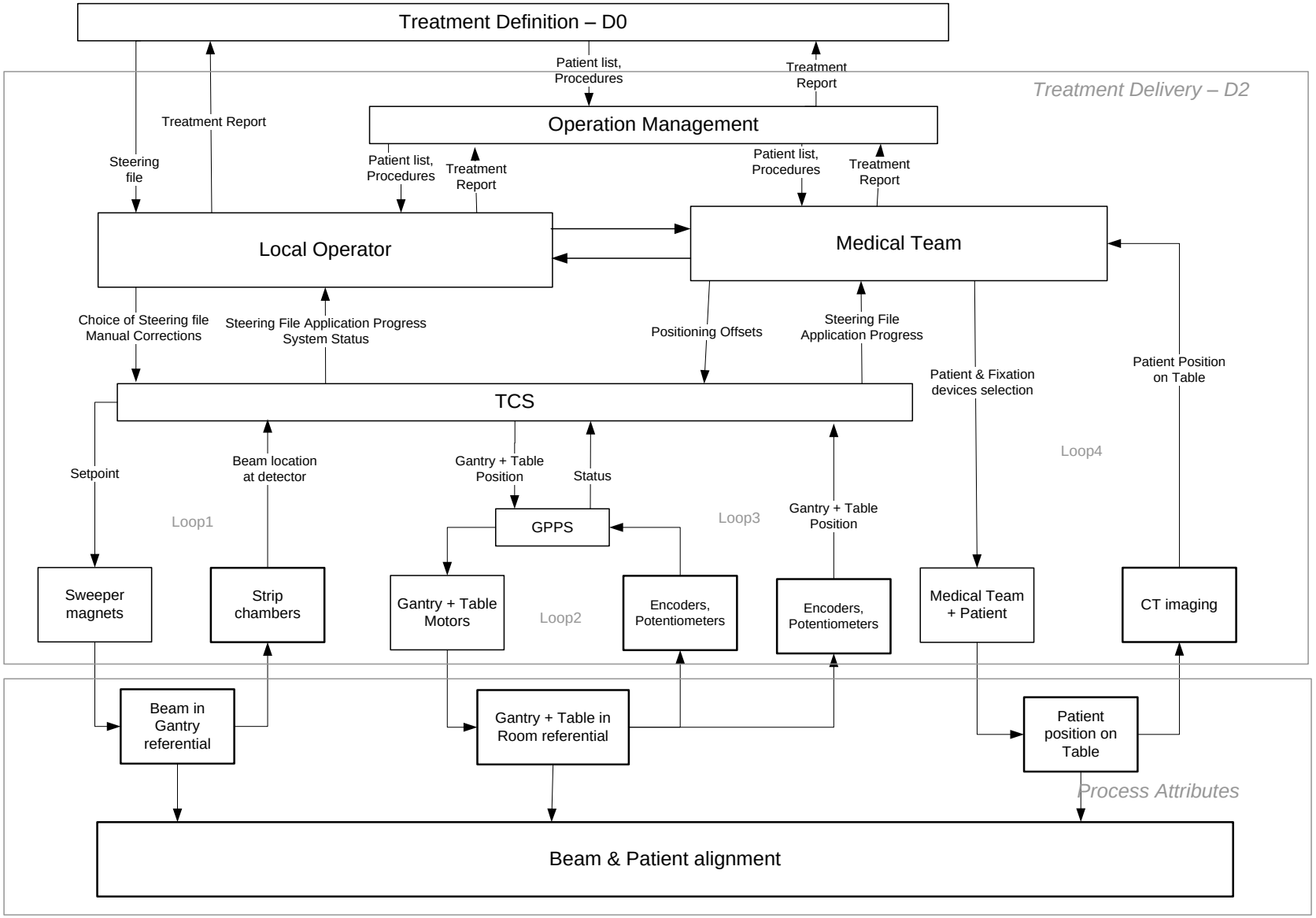


Figure 2.11. A more detailed view of Treatment Delivery

Once the basic structure is defined, the additional details about the control structure need to be added, that is, the responsibilities and process model for each controller, the control actions, and the feedback. The responsibilities are the basic high-level requirements for the components. As the analysis progresses, the controller safety-related responsibilities and requirements will be identified. There may also be communication between controllers, and this communication needs to be shown. Figure 2.12 shows a more complete control structure for an automated train door controller.

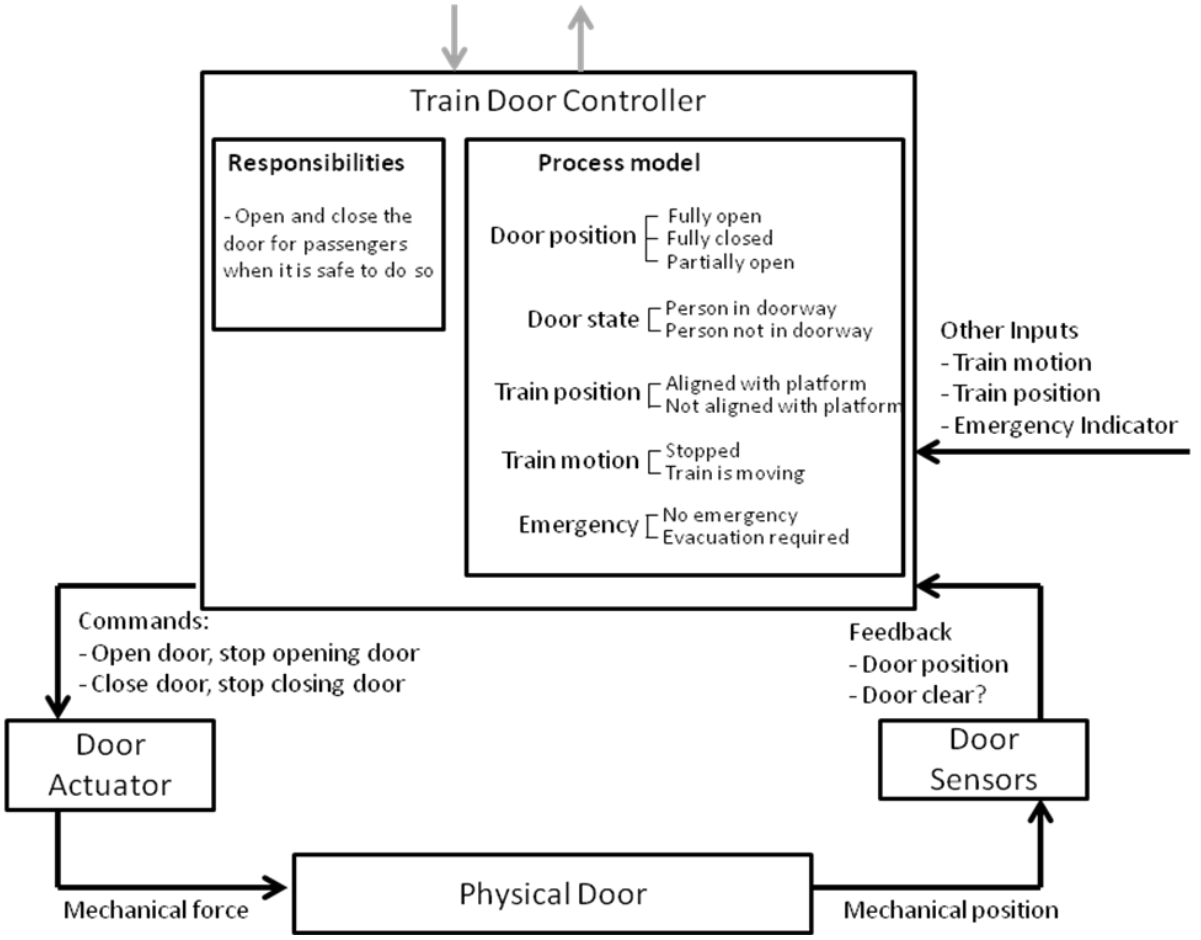


Figure 2.12. Simple Safety Control Loop for a Train Door Controller

In our experience modeling and doing STPA analyses, we have often found copious documentation on the physical structure of a system but much less about the functional design. In all these cases, the documentation of the functional control structure we created was greatly appreciated by others working on the system.

> **Exercise**: Create the functional safety control structure for the batch reactor system shown on page 9 of ESW. What is an accident in this system? What is the system-level safety constraint involved in the accident?

FAQ: Does the control structure always have to be hierarchically linear, like a ladder?

No. The control structures in this primer are only examples from various applications, and yours may be different. Although some control structures look like a ladder, like the generic operations and development structure shown in Figure 1.5 or the simple Aircraft -> Pilots -> ATC structure for the NextGen ITP example in Figure 2.2, this is not always the case. For example, the high-level control structure for the Gantry 2 radiation therapy machine shown in Figure 2.9 shows how control and feedback paths can skip across levels in the control structure. Also, the detailed control structures for the same system show how multiple controllers can operate at the same level, as with the medical team and the local operator in Figure 2.11. There is no fixed structure that must be used for all systems.

Identifying Unsafe Control Actions (STPA Step 1)

While it is convenient to separate STPA into two steps, first identifying the unsafe control actions and then the causes of the unsafe control actions, this separation is not necessary. The two steps could be integrated in various ways, for example, identifying an unsafe control action and immediately looking for its causes.

The four types of unsafe control action described in Chapter 1 are:

- A control action required for safety is not provided
- An unsafe control action is provided that leads to a hazard
- A potentially safe control action provided too late, too early, or out of sequence
- A safe control action is stopped too soon or applied too long (for a continuous or non-discrete control action)

There is a fifth way that safety can be compromised—a required control action is provided but is not followed— but that fifth possibility will be handled in STPA Step 2.

We have found that a table is a convenient way to document the specific unsafe control actions but any format could be used. The general form of the table that we use is:

Control Action	Not providing causes hazard	Providing causes hazard	Too early/too late, wrong order causes hazard	Stopping too soon/applying too long causes hazard

Continuing with ITP as the example and using the hazard "Loss of minimum separation," Table 3 might be the result for the flight crew unsafe control actions. If there are multiple hazards being analyzed, either different tables might be used or the entries in the table might indicate (point to) the hazard or hazards involved. If a box in the table is empty, then that control action cannot be unsafe. We have found that there may be duplicates entries in the table in the sense that they are semantically equivalent. For example, in Table 3, "ITP executed with incorrect final attitude" in row 1, column 3 is the same as the entries in the last column in that row. Either the duplications can be omitted or they can be left and only one of the duplicates used in the following steps of the analysis. STPA Step 1 is only a part of the process; it is not the final answer.

Notice that the identified unsafe control actions in the tables all have *conditions* or context associated with them. If executing the control action is always unsafe, then it would make no sense to include it in the system design. Almost always, there are only some conditions under which the control actions will be

unsafe, and the goal of Step 1 is to identify those. Executing an ITP is not always unsafe; it is only unsafe when the ITP execution has not been approved by ATC, when the criteria are not satisfied, etc.

Table 3: Unsafe flight crew control actions for the hazard Loss of Minimum Separation for ITP

Hazard: Loss of minimum separation				
Control Action	**Not Providing CA Causes Hazard**	**Providing CA Causes Hazard**	**Wrong Timing/Order of CA Causes Hazard**	**CA Stopped Too Soon/Applied Too Long**
Execute ITP		ITP executed when not approved ITP executed when ITP criteria are not satisfied ITP executed with incorrect climb rate, final altitude, etc.	ITP executed too soon before approval ITP executed too late after reassessment	ITP aircraft levels off above requested FL ITP aircraft levels off below requested FL
Abnormal Termination of ITP	FC continues with maneuver in dangerous situation	FC aborts unnecessarily FC does not follow regional contingency procedures while aborting		

Table 4: Unsafe control actions for the air traffic controller.

Hazard: Loss of minimum separation				
Control Action	**Not Providing CA Causes Hazard**	**Providing CA Causes Hazard**	**Wrong Timing/Order of CA Causes Hazard**	**CA Stopped Too Soon or Applied Too Long**
Approve ITP request		Approval given when criteria are not met Approval given to incorrect aircraft	Approval given too early Approval given too late	
Deny ITP request				
Abnormal Termination Instruction	Aircraft should abort but instruction not given	Abort instruction given when abort is not necessary	Abort instruction given too late	

The entries in the table can be translated into safety constraints/requirements on the component considered in the table. For example, the safety constraints the flight crew must implement are:

SC-FC.1 The flight crew must not execute the ITP when it has not been approved by ATC.

SC-FC.2 The flight crew must not execute an ITP when the ITP criteria are not satisfied.

SC-FC.3 The flight crew must execute the ITP with correct climb rate, flight levels, Mach number, and other associated performance criteria.

SC-FC.4 The flight crew must not continue the ITP maneuver when it would be dangerous to do so.

SC-FC.5 The flight crew must not abort the ITP unnecessarily. (Rationale: An abort may violate separation minimums)

SC-FC.6 When performing an abort, the flight crew must follow regional contingency procedures.

SC-FC.7 The flight crew must not execute the ITP before approval by ATC.

SC-FC.8 The flight crew must execute the ITP immediately when approved unless it would be dangerous to do so.

SC-FC.9 The crew shall be given positive notification of arrival at the requested FL

Similar safety constraints on ATC can be generated from Table 4. Notice what has occurred here. The very high-level safety constraint/requirement derived from the loss of separation hazard, i.e., "The ITP maneuver must never result in a loss of safe separation between aircraft," has now been refined and allocated to the system components. STPA Step 2 will refine and add additional component safety requirements. As the design process proceeds and more detailed design decisions are made, STPA is used to create even more detailed requirements.

STPA is a top-down, system engineering process to create a safe system design and not just an after-the-fact method to analyze a completed design. As such, STPA can be started very early in the system engineering life cycle, even in the concept development stage. Most other existing hazard analysis methods require that a concrete design exists before they can be used. Finding out that designs have safety flaws in them late in the review process can lead to extremely costly rework, especially if the requirements turn out to be wrong, which is almost always the case for software that is involved in accidents. The cost of adding STPA from the start to the system engineering process is negligible.

> **Exercise:** Take the control structure you created for the batch reactor in an earlier exercise and create the Step 1 tables. Then change the entries into requirements for the operator and the computer software.

On the next few pages, two examples of Step 1 tables (and different potential formats) are shown, the HTV docking procedure and closing a critical valve in the nuclear power plant example.

#	Control Action	Not Providing Causes Hazard	Providing Causes Hazard	Wrong Timing/Order Causes Hazard	Stopping Too Soon /Applying Too Long Causes Hazard
1	*FRGF Separation Enable*	**[UCA1]** FRGF separation is not enabled when ready for capture	**[UCA2]** FRGF separation is enabled when not necessary (e.g. after successful capture)	EARLY: **[UCA3]** FRGF separation is enabled while not ready for immediate capture	
2	*Free Drift* (Deactivation)	**[UCA4]** HTV is not deactivated when ready for capture	**[UCA5]** HTV is deactivated when not appropriate (e.g., while still approaching ISS)	EARLY: **[UCA6]** HTV is deactivated while not ready for immediate capture LATE: **[UCA7]** HTV is not deactivated for a long time while FRGF separation is enabled	
C	Execute Capture	**[UCA8]** Capture is not executed while HTV is deactivated	**[UCA9]** Capture is attempted when HTV is not deactivated **[UCA10]** SSRMS hits HTV inadvertently	EARLY: **[UCA11]** Capture is executed before HTV is deactivated LATE: **[UCA12]** Capture is not executed within a certain amount of time	**[UCA13]** Capture operation is stopped halfway and not completed
3	*FRGF Separation Inhibit*	**[UCA14]** FRGF separation is not inhibited after successful capture	**[UCA15]** FRGF separation is inhibited when must be enabled (e.g., when capture is attempted)	LATE: **[UCA16]** FRGF separation is inhibited too late after successful capture	
Off-Nominal	*Abort Retreat Hold*	**[UCA17]** Abort/Retreat/Hold is not executed when necessary (e.g., when HTV is drifting to ISS while uncontrolled)	**[UCA18]** Abort/Retreat/Hold is executed when not appropriate (e.g. after successful capture)	LATE: **[UCA19]** Abort/Retreat/Hold is executed too late when immediately necessary (e.g., when HTV is drifting to ISS while uncontrolled)	
	FRGF Separation	**[UCA20]** FRGF separation is not executed when necessary (e.g., when HTV is grappled unsafely)	**[UCA21]** FRGF separation is executed when not necessary (e.g., after successful capture)	LATE: **[UCA22]** FRGF separation is executed too late when immediately necessary (e.g., when HTV is grappled unsafely)	

Accident		Hazard		UCA
A1	Collision with ISS	H1	HTV is drifting to ISS while uncontrolled (deactivated)	5, 6, 8, 12, 17, 19
		H2	HTV is unintendedly separated from SSRMS after successful capture	2, 14, 16, 21
A2	Damage to SSRMS	H3	HTV provides unintended attitude control in proximity to SSRMS	4, 9, 11
		H4	HTV is inclined by a large angle in proximity to SSRMS	10
		H5	HTV cannot be separated immediately when grappled unsafely (e.g., windmill)	1, 13, 15, 20, 22
		H6	HTV provides thrust while captured by SSRMS	18, 20, 22
A3	Loss of HTV mission	H7	FRGF is unintendedly separated from HTV before or during capture	2, 3, 7, 21

Control Action	Unsafe 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

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

FAQ: How do I know if the entries in the tables are complete?

When we first started generating these tables, we relied on expert review to identify missing or incorrect entries. Using the tables helped with this review process. Since that time, John Thomas created a formal process for generating the tables (described in Chapter 3). When we went back and applied it to earlier tables, we invariably found missing entries.

In general, within itself, the Thomas process can be shown to be complete, that is, it will identify all the unsafe control actions for the conditions considered. However, it will not be complete if the engineers omit from consideration (either purposely or accidentally) some conditions that are in fact important or the humans err in making a final determination of whether the control action under those conditions is unsafe or not. The first case is less likely than the second. One of the advantages of the Thomas method for generating the unsafe control actions is that much of it can be automated. Chapter 3 contains more information about the Thomas method and how to use it.

FAQ: Why aren't there any component failures in these tables? Does that mean STPA ignores failures?

These tables are only Step 1 of STPA. They describe unsafe behavior (in terms of unsafe control actions) in the system. Potential causes of this behavior, which could include physical failures, human error, design errors, or requirements flaws, are part of STPA Step 2 and are not included in these Step 1 tables. This two-step process helps ensure that the STPA analysis is efficient and no time is wasted considering failures or errors that lead to *safe* behavior or have no effect, as in a FMECA.

Stopping after Step 1 can lead to omissions in the requirements and unsafe design. Step 2, which identifies the causes of the unsafe control actions and also the causes of why required control actions might be implemented correctly, identifies more requirements and provides assistance to the engineer in identifying changes in the design to eliminate or mitigate unsafe control.

Identifying the Causes of the Unsafe Control Actions (STPA Step 2)

Once the safety control actions are identified (or once any of the unsafe control actions are identified, i.e., the process does not have to be completely serial), the second and final step in STPA is to identify the potential causes of (scenarios leading to) unsafe control. Here is where the fifth type of scenario, inadequate execution of a control action required for safety, is considered.

Step 2 requires the most thought and prior experience by the analyst and there is, so far, much less help provided compared to Step 1. Therefore, we have found that sometimes STPA is stopped after Step 1. Step 2 is critical, however, as you will see in the first exercise. Step 2 identifies additional safety requirements both on the controller in the loop being analyzed and on the overall system. It is also where information is generated to assist the designers in eliminating or mitigating the potential causes of the hazards. The most important reason to do a hazard analysis at all is to get the causal information generated by Step 2.

Basically, the Step 2 process involves examining the control loop and its parts and identifying how they could lead to unsafe control. Figure 2.13 shows things that can go wrong in the control loop. Each of these is discussed in Chapter 4 of ESW.

Care should be taken here to not turn this step into a form of FMEA by simply looking at each of the "guidewords" in Figure 2.13 and seeing whether they lead to the hazard. The goal is not to find just failures or inadequate operation of individual components in the control loop, but to find scenarios and combinations of problems that could lead to unsafe control. The process should be to start with the unsafe control actions and determine how they could occur as well as how actions required for safety might not be executed correctly.

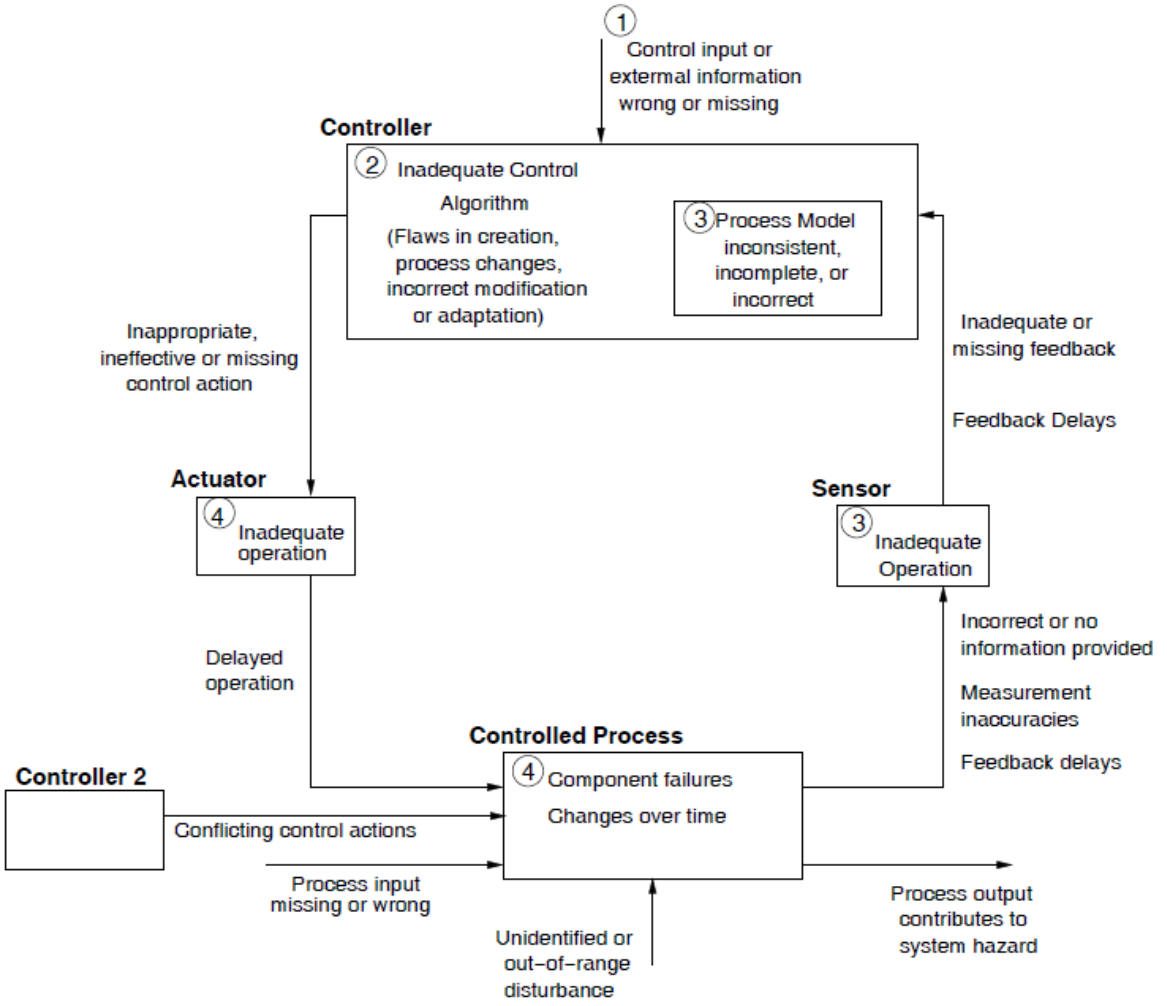


Figure 2.13 Things that can go wrong in the control loop

Because there are common flaws that lead to accidents, we hope to provide more assistance for Step 2 in the future, some of which might be able to be supported by automation. Step 2 is a good place for small groups of engineers to work together brainstorming causes. Step 1 can be accomplished by a single person with later review by others. But identifying causes is enhanced by having multiple people participating.

Usually the cause of the generation of an unsafe control action can be found in the right side of the loop while the cause of not executing a control action or not executing it adequately is in the left side but this rule is not always true. See Figure 2.14.

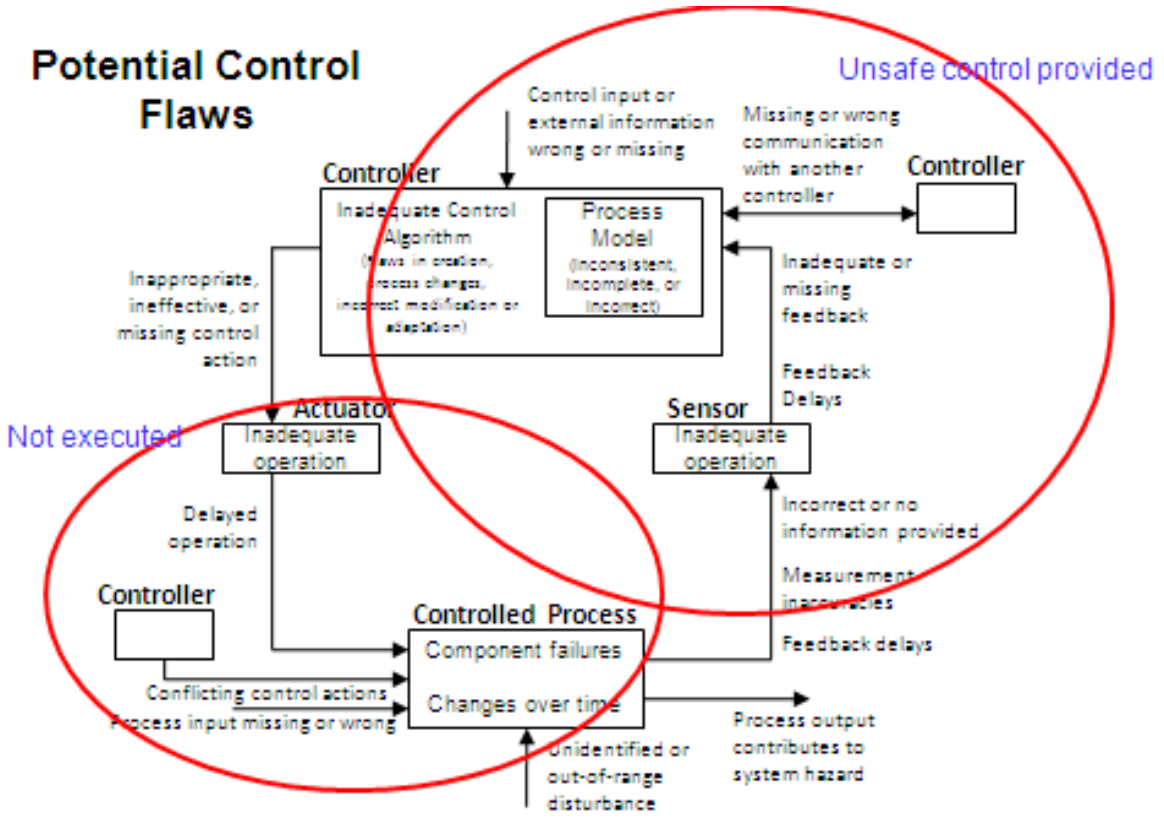


Figure 2.14. Potential control flaws related to parts of the control loop.

Delays in the loop are an important consideration in the causal analysis. Loop delays can lead to process models being (temporarily) inconsistent with the process and thus to the controller providing unsafe control. There may also be problems in updating the process model. Chapter 9 of ESW provides some common design flaws that can be considered in Step 2. I am a little concerned about making a "checklist" of things to consider, however, as that can cause more problems than it solves by limiting what is considered during the causal analysis. Checklists tend to limit consideration to those things that are in the checklist. Even the guidewords in Figure X tend to focus attention on them to the exclusion of other, perhaps more obscure or less common factors.

Exercise: Take your Step 1 table that you generated for the batch chemical reactor and identify causal scenarios for the unsafe control actions. At the least,

- Identify some causes of the hazardous control action: *Open catalyst valve when water valve is not open.* HINT: Consider how controller's process model could identify that the water valve is open when it is not.
- What are some causes for a required control action (e.g., open water valve) being given by the software but not executed?
- What design features (controls) might you use to protect the system from the scenarios you found?

We usually use a table, lists, or annotated control loop diagrams to document the Step 2 results. Understanding the scenarios by providing graphical depictions of the states of the system leading up to the hazard has also been helpful. For example, the relative positions of aircraft might be shown preceding and

leading up to a hazardous state (e.g., loss of minimum separation) at each critical point in the scenario for the hazard.

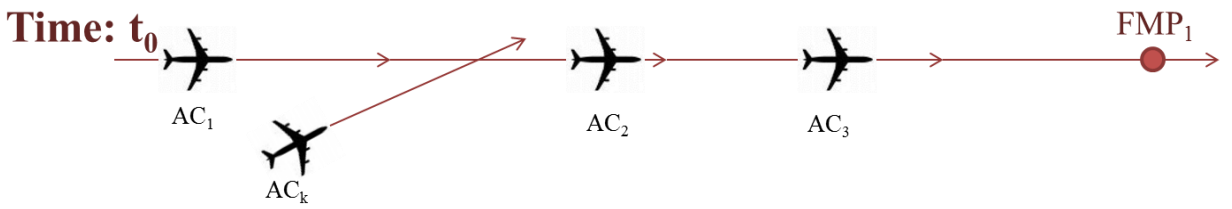


Figure15

(a)

Figure15 (b)

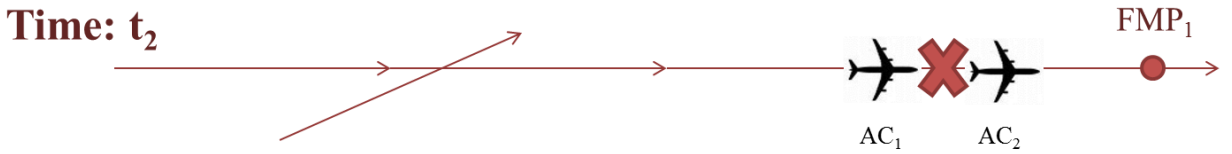


Figure 15 (c)

Figure 2.15. Example Scenario for Loss of Separation (UCA.6.S)

Industry-specific (and perhaps application-specific) tools to help depict the scenarios could be immensely useful to designers.

The following Table shows a small part of the causal analysis (associated with a process model flaw) for an unsafe control action in a separation assurance task (GIM-S) for an air traffic control system. The system includes a human controller and an automated Traffic Flow Manager (TFM), i.e., two controllers over the same process. Only one unsafe control action is considered in the table: *Air Traffic Controller providing a speed modification to an aircraft too late after a different clearance has been executed by the same or a different aircraft (AC) by the flight crew (FC).* Also, only the process model flaws leading to the unsafe control action are included here and not other causes.

UCA: Air Traffic Controller provides a speed modification to an aircraft too late after a different clearance has been executed by the same or a different aircraft (AC)		
Scenario	**Associated Causal Factors**	**Rationale/Notes**
[Process Model Flaw: Aircraft / FC Model] ATC is unaware of another clearance the Aircraft/FC is executing or has requested.	FC or AC not flying the flight plan visible to the ATC	FC and AC will require their own detailed hazard analysis.
	Aircraft has recently been passed to a new controller (sector or shift change) and the in-process	

UCA: Air Traffic Controller provides a speed modification to an aircraft too late after a different clearance has been executed by the same or a different aircraft (AC)		
Scenario	**Associated Causal Factors**	**Rationale/Notes**
	clearance was not conveyed	
	Other clearance for same aircraft was not entered into TFM automation upon issuance or execution	Another ATC issues a clearance on that is not part of GIM-S and thus not given to the automation
	Center TFM update rate is too slow, new trajectory is not computed until after different clearance is issued (external input flaw)	Computation of new advisory takes too long, or specified refresh rate is too slow
	Signal gets jammed, corrupted	This could be corruption of signal between TFM and ATC or between ATC and FC
	AC position or speed is incorrect due to surveillance delay so ATC is not aware of mismatch	
[Process Model Flaw: Airspace] ATC is unaware a clearance being issued to another aircraft.	Clearance for another aircraft was not entered into TFM automation [process model not updated correctly]	Another ATC accepted an advisory but did not indicate so in the TFM automation. OR ATC issued an advisory not from automation without updating flight plan
[Process Model Flaw: Airspace – predicted separation] TFM and/or ATC is not aware of how changing environmental conditions affect prior clearances	Trajectory information is incorrect because dead-reckoning or other predictive strategy has incorrect or insufficient wind data	
	Strategies and trajectories are modified due to the presence or prediction of convective weather	TFM automation receives weather data that ATC does not see or have access to
		ATC or FC 'sees' convective weather that does not go into TFM model
[Process Model Flaw: Airspace – sequence & flow] ATC prioritizes issuing clearance to another aircraft	Conflict involving the other aircraft is imminent and requires immediate action	Conflict receives priority (correctly)
	Clearance in conflict with onboard RA	TCAS or other advisory is different than clearance provided for GIM-S
Process Model Flaw: Model of TFM Automation]		
	TFM model of airspace is different than ATC model due to missing feedback or input	Lack of surveillance information for TFM, no ADS-B or mixed equipage

UCA: Air Traffic Controller provides a speed modification to an aircraft too late after a different clearance has been executed by the same or a different aircraft (AC)		
Scenario	**Associated Causal Factors**	**Rationale/Notes**
		ATC or other entity does not update flight plans (see previous casual factors)
	Modified flight plans are not input into TFM trajectory model	Flight plans might be modified by operators prior to flight or during flight, and mods are simply not given
	Flight plans are input incorrectly into TFM automation	Incorrect format or updated too late

Another potential format for the causal scenario information follows.

Unsafe Control Action for EnRoute ATC: Provide a speed modification to an a/c too late after a different clearance has been executed by same or other a/c.

Scenario 1: ATC is unaware of another clearance the AC/FC is executing or has requested [ATC process model of AC/FC incorrect]

a) FC/AC not flying flight plan visible to ATC
b) AC/FC has recently passed to a new controller (sector or shift change) and in-process clearance was not conveyed.
c) Other clearance for same aircraft was not entered into TFM automation upon issuance or execution.
d) Center TFM update rate too slow. New trajectory is not computed until after different clearance is issued.
e) Signal gets corrupted or jammed.
f) AC position/speed is incorrect due to surveillance delay so ATC is not aware of mismatch.

Scenario 2: ATC is unaware of clearance being issued to another aircraft because clearance not entered into TFM.

a) Another ATC accepted an advisory but did not indicate so in TFM.
b) ATC issued an advisory not provided by the automation and did not update the flight plan.

Scenario 3: Airspace predicted separation is incorrect in ATC or AFM.

a) Trajectory information is incorrect because dead reckoning or other predictive strategy has incorrect or insufficient wind data.
b) Strategies and trajectories are modified due to presence or prediction of convective weather.
    i. TFM automation receives weather data that ATC does not see or have access to.
    ii. ATC or FC "sees" convective weather that does not go into TFM.

Once the causal scenarios are identified, they can be used to provide detailed requirements for the designers in order to avoid the hazard. The table below shows a small part of a table showing the scenario, the associated causal factors along with requirements generated to eliminate or mitigate the causal factors as well as the system component to which the requirements will be allocated.

Scenario	**Associated Causal Factors**	**Requirement**	**Allocated To**	**Rationale**
STPA-F.14M.1 [Process Model Flaw: Aircraft / FC Model] ATC believes that FC is (or will be) flying a different speed, therefore ATC assumes that separation requirements will be met, and/or issues other clearances based on this assumption.	STPA-F.14M.1.1 Incorrect aircraft ID on radar or flight strip	STPA-F.14M.1.1.1 Modified flight plans or new clearances must be sent to FIM automation within TBD seconds for all aircraft in sector	Operators, Controllers	Modified plan in fleet by operator
		STPA-F.14M.1.1.2 The design of user interfaces must minimize incorrect inputs.	ERAM, FIM Automation, Other ATC or Operator Interfaces	Incorrect input into display by either ATC, FC, or operator
		STPA-F.14M.1.1.3 User interfaces must provide a clear, consistent means for entering aircraft data.	ERAM, FIM Automation, Other ATC or Operator Interfaces	
		STPA-F.14M.1.1.4 Airline operator must verify that the registration/call sign matches the associated aircraft data file	Airline operators	

As an example of another possible format, the following shows part of the causal analysis and possible controls that might be used for the Gantry 2 proton radiation therapy system (Figures 2.7 through 2.12):

- **Scenario 1**: Operator was expecting patient to have been positioned, but table positioning was delayed compared to plan because of
  - Delays in patient preparation
  - Delays in patient transfer to treatment area
  - Unexpected delays in beam availability
  - Technical issues being processed by other personnel without proper communication with operator
  - …

  Controls:
  - Provide operator with direct visual feedback to the gantry coupling point and require check that patient has been positioned before starting treatment
  - Provide a physical interlock that prevents beam on unless table positioned according to the plan

- **Scenario 2**: Operator is asked to turn the beam on outside of a treatment sequence because
  - Design team wants to trouble shoot a problem
  - …

  but inadvertently starts treatment and does not realize that the facility proceeds with reading the treatment plan.

  Controls:
  - Reduce the likelihood that non-treatment activities have access to treatment related input by creating a non-treatment mode to be used for QA and experiments, during which the facility does not read treatment plans that may have been previously loaded;
  - Make procedures to start treatment sufficiently different from non-treatment beam-on procedures that the confusion is unlikely.

FAQ: If STPA is an iterative, refinement process, how do I know when I can stop or do I have to go on forever?

In the top-down STPA analysis approach, the analyst can stop refining causes at the point where an effective mitigation can be identified and not go down any further in detail. The analyst only has to continue refining causes if an acceptable mitigation cannot be designed. That is the major difference between STPA and bottom-up techniques like FMEA and one reason why FMEA takes more effort and resources for a complex system than does STPA.

Using STPA for Preliminary Hazard Analysis (PHA): Cody Fleming

To be completed in the future.

Applying STPA to Management, Social Systems, and Project Risk Analysis: John Helferich

To be completed in the future.

Extensions to STPA to Include Advanced Human Factors Concepts

The STPA process provides more information to system designers and evaluators about the role of human errors in hazard causation by going beyond treating human error as random failure. In STPA as now defined, human error is treated in a similar way as software error. This makes sense as most software is introduced to replace human operators, but there are important differences in how a human behaves and how computers behave and we are exploring how to incorporate these into STPA. When we are confident that our procedures are effective and useful, we will provide detailed information in this chapter.

Chapter 3: Formal Tools to Support STPA

John Thomas

(First version: September 2013
Change history:)

This chapter describes a more systematic method for performing STPA Step 1 and identifying Unsafe Control Actions (UCAs). The systematic method has been useful as a way to provide more guidance to people who are new to STPA, to ensure consistency in the way UCAs are written, and as a way to help identify any UCAs that might have been overlooked using ad-hoc methods. This chapter summarizes the systematic method, but for a more in-depth discussion and more detailed examples see the recent Ph.D. dissertation by John Thomas [Thomas 2013]. The dissertation also provides more information about automating parts of the process as well as requirements generation directly from the results of this analysis.

The Main Elements of an Unsafe Control Action

Recall that control actions are commands that are sent by a controller to control some process. When control actions are inadequate and lead to a hazard, they are considered Unsafe Control Actions. The key to using the systematic method for STPA Step 1 is to recognize that a control action is not hazardous by itself. For example, consider a train door controller for the Boston subway system. Is the control action *open train doors* safe or unsafe? The answer is that it depends. To figure out whether an action is safe or unsafe, we need to define the context. *Open train doors <u>while train is moving</u>* would be an unsafe control action, but *open train doors <u>while stopped at a platform</u>* is not only a safe control action, it is one that is required for proper system behavior.

Notice in this example that the context is really part of the controller's process model[4]. This is by necessity because the controller cannot make safe decisions unless the controller can somehow distinguish between safe and unsafe contexts for the control actions. This is true for software control actions as well as for human control actions, and in fact the door controller could be implemented either way. Because STPA is a functional analysis, implementation details like these do not need to be known immediately in order to get useful results. In fact, the most efficient way to use STPA and the systematic method in this chapter is to apply them in parallel with the design process. Each iteration of STPA produces more refined requirements to drive the design and each iteration of the design produces information that refines the STPA analysis.

In the train door example, the contexts mentioned above could be expressed in a more familiar format as process model variables:

[4] As you may recall, the controller's process model or mental model essentially captures the controller's beliefs about the outside world. The model is updated by feedback and it is used by the controller to help decide what control actions are needed.

Train motion

- Moving
- Stopped

Train location

- At a platform
- Not at a platform

Obviously there are other control actions and process model variables for the train door controller, but the point here is to show that the context of UCAs must be included in the controller's process model, and both define the information the controller needs in order to make safe decisions.

Figure 3.1 defines four elements that make up a UCA. The first element is the Source, which is the controller providing the action and can be obtained from the control structure. The second element is the Type, which could be *provided* or *not provided*—either of which may be hazardous in different contexts. The third element is the name of the control action itself and can be obtained from the control structure. The last element is the context. The systematic method essentially identifies each potential element from the control structure and other sources and then considers how multiple elements can combine to form UCAs.

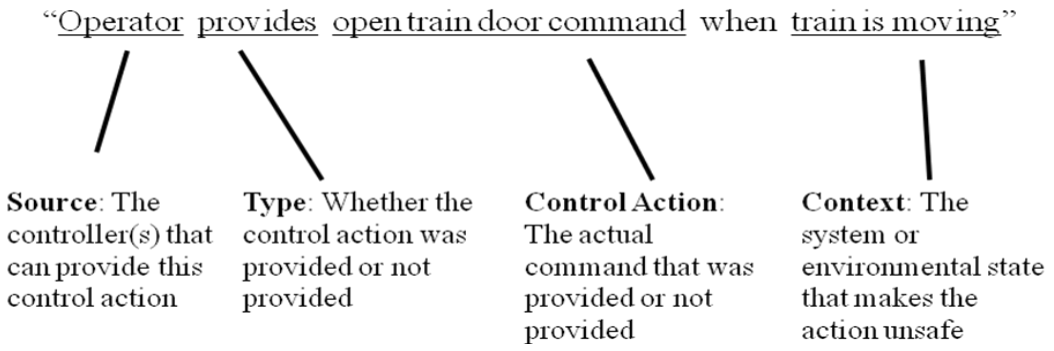


Figure 3.1: The Structure of an Unsafe Control Action

The Systematic Method

The systematic method starts by selecting a controller and control action from the control structure and constructing a *context table* as shown in Table 3.1. The first column indicates that this table analyzes the control action *Door Open*. The next three columns correspond to the process model variables for the selected control action. Each row is populated with a unique combination of process model values, i.e., a unique context.

There are two parts to the systematic method, and each part can be performed independently of the other. The first part analyzes control actions that are provided under conditions that make the action hazardous. The second part analyzes control actions that are *not* provided under conditions that make *inaction* hazardous.

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 3.2.

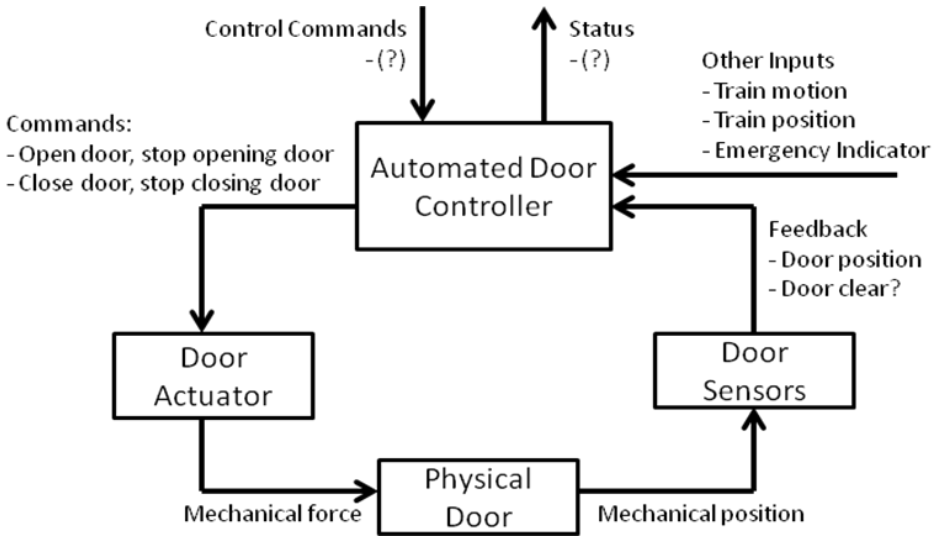


Figure 3.2: Partial control structure for simplified train door controller

Part 1: Control actions provided in a state for which 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 door controller can provide four control actions: *open doors, stop opening doors, close doors, or stop closing doors*.[5] 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. The required variables in the process model can be derived from the system hazards defined at the start of an STPA analysis, from the required feedback in the control structure, and from other knowledge of the environmental and process states. For example, hazard H-1 in the train door example indicates that the state of the doorway (whether it is clear or not) is an important environmental variable in deciding whether to close the doors. Figure 3.3 shows the control structure including the required process model for the door controller.

[5] 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 other words, the functional commands themselves are analyzed independently of whether they are ultimately implemented as continuous or discrete signals.

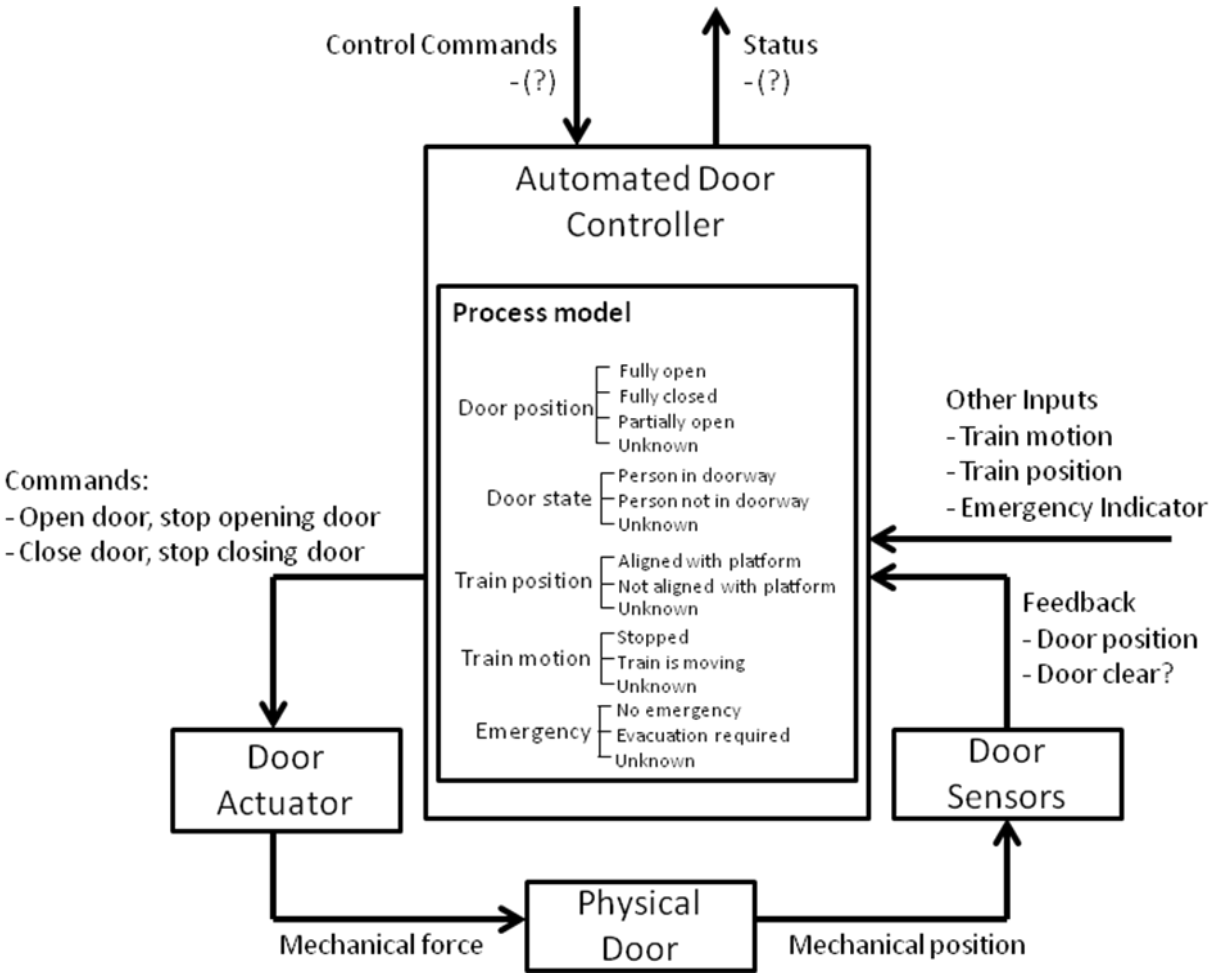


Figure 3.3: Augmented control structure with the door controller's process model

Exercise: Identify the process model variables for the software controller in the batch reactor system from the previous chapter.

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

Each row in Table 3.1 specifies a different context for the *open door* 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. 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 3.1: Context table for the *open door* control action

Control Action	Train Motion	Emergency	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 (H-2)	Yes (H-2)	Yes (H-2)
Door open command provided	Train is moving	Emergency exists	(doesn't matter)	Yes[6](H-2)	Yes (H-2)	Yes (H-2)
Door open command provided	Train is stopped	Emergency exists	(doesn't matter)	No	No	Yes (H-3)
Door open command provided	Train is stopped	No emergency	Not aligned with platform	Yes (H-2)	Yes (H-2)	Yes (H-2)
Door open command provided	Train is stopped	No emergency	Aligned with platform	No	No	No

Exercise: Using the process model variables you identified for the batch reactor software, create a context table like Table 3.1 for the Open Catalyst Valve control action.

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 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. This is an example of how STPA can be applied during early design phases to help engineers uncover and resolve conflicts as early as possible when the most important design decisions are not yet set in stone.

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 3.2 shows the identification of unsafe control actions for the door open command not being provided.

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

Table 3.2: 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[7]
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 (H-1)
Door open command not provided	Train is stopped	No emergency	Not aligned with platform	Person in doorway	No[8]
Door open command not provided	Train is stopped	Emergency exists	(doesn't matter)	(doesn't matter)	Yes (H-3)
Door open command not provided	Train is moving	(doesn't matter)	(doesn't matter)	(doesn't matter)	No

Exercise: Using the process model variables you identified for the batch reactor software, create a context table like Table 3.2 for the lack of an Open Catalyst Valve control action.

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 still 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.

FAQ: Some of the contexts are hazardous by themselves, like a person in the doorway while the train is moving. Should the final column always be marked hazardous in these cases?

No. Contexts that are already hazardous by themselves should of course be avoided or made impossible by design, but that is not always feasible. If the system ever gets into a hazardous state, the controllers must provide appropriate control actions to return the system to a safe state.

[7] 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.

[8] 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.

Therefore, even if the context is already hazardous by itself, these tables need to define whether the action will keep the system in a hazardous state or will return the system to a safe state.

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 3.3.

Table 3.3: 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 (H-1) Door open command not provided when train is stopped and emergency exists (H-3)	Door open command provided when train is moving and there is no emergency (H-2) Door open command provided when train is moving and there is an emergency[9] (H-2) Door open command provided when train is stopped unaligned with platform and there is no emergency (H-2)	Door open command is provided more than X seconds after train stops during an emergency (H-3)	N/A

Notice that this approach documents traceability from each UCA back up to the system-level hazards defined at the start. When requirements are generated, this ensures that each refinement is traceable all the way from the highest level requirement down to the lowest level.

[9] 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.

Manual and automated techniques for complex applications

The procedures described here represent a "brute-force" approach and, although it is relatively easy to follow and provides a more systematic approach with more guidance, it can be time consuming when applied to low-level contexts with many variables and values. To address this issue, there are a number of automated algorithms that can be used to assist in the analysis as well as both manual and automated techniques that can be employed to reduce the amount of effort required to analyze extremely complex systems. These topics are discussed in this section, and several tools are currently in development based on these techniques.

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 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 the context table 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 in a top-down fashion before that level of detail is defined.

In fact, an important goal of this approach is to help during 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.

Logical simplification

The second technique—*logical simplification*—was already employed when introducing the context tables for the train door. In this example, the four columns of variables each with two possible values would really require a 16 row table. However, the context table in Table 3.2 only required six rows. By reducing similar rows with "doesn't matter" terms, the table can be drastically simplified. For example, the last row in Table 3.2 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.

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

Continuous process model variables

The context table examples provided above 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 hazard is simply whether the train is stopped (velocity equals zero) or moving (velocity not equal to zero). Through careful discretization of process variables based on the system hazards, the complexity of the context tables and subsequent analysis can be significantly reduced. In addition, automated tools discussed later in this section 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 this section 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.

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, 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, a rule might be defined that states that opening the doors while the train is moving will always lead to H-2, and the tool can automatically populate the final column for those 8 rows of the context table with the hazard H-2.

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. 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 a potentially incomplete set of rules).

Although these concepts 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 that follows is a much more practical way to include such low levels of detail.

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. The following is a technique that can be used to generate extremely detailed context tables from more abstract tables and information.

Consider the high-level context table for the train door controller, reproduced in Table 3.4. 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 3.4 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 3.4: 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[10]
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[11]
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:

[10] 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.

[11] 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 3.4 could be defined. SpecTRM-RL tables (also known as AND/OR tables) are a disjoint form of Boolean logic. They are read with the rows using AND logic and the columns using OR logic. In other words, the box to the left contains expressions while the narrow vertical columns show the conditions under which the expression about the table is true. If any one of the vertical columns is true, then the statement about the whole table is true. In the first example below, the **Door_obstructed** variable is inferred to have the value **obstructed** when the light curtain is blocked or the door force sensor shows that the door has been pushed open. The **Door_obstructed** variable is inferred to have the value **not obstructed** if the light curtain is not blocked ***and*** the door force sensor shows a normal value.

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 3.4: 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 3.4. Table 3.5 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 3.5: 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 automated methods. By applying logical simplification techniques, requirements even for larger tables as in Table 3.5 can be automatically reduced to an equivalent but much smaller form. Figure 3.5 shows the logically simplified SpecTRM-RL table that is generated based on the high-level context tables (as in Table 3.5) and defined relationships between process model variables (as in Figure 3.4).

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

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. A number of tools are possible, 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 can 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 and existing software tools like SpecTRM that help document traceability and document rationale behind decisions.

Chapter 4: Evaluation of STPA on Real Systems

Nancy Leveson

[*Because more experience is being obtained about the use of STPA on various types of systems and additional comparative studies conducted, this chapter will be updated as we get more information.*]

Because STAMP extends current accident models and thus includes component failure accidents, STPA can identify the hazard scenarios identified by fault tree, event tree, and other traditional hazard analysis methods, but it also can find those factors not included or poorly handled in these traditional methods such as software requirements errors, component interaction accidents, complex human errors (mistakes vs. slips), inadequate coordination among multiple control agents, and unsafe management and regulatory decision making.

While this comparison of STPA and the traditional hazard analysis methods shows STPA to be *theoretically* more powerful, does STPA actually identify more causal scenarios when used on real systems? There have been a lot of real-world comparisons made and in each of these STPA outperformed the traditional hazard analysis methods.

One of the first industrial uses of STPA, in 2003, was on the new U.S. Missile Defense System in order to assess the risk associated with the hazard of inadvertent launch [Pereira 2006]. The system had been subjected to standard hazard analysis methods, but one more additional analysis was required before the system could be deployed and field tested. STPA found so many flaws during just a limited three month analysis by two people that deployment was delayed for six months to fix the newly identified hazardous scenarios. In many of these newly identified scenarios, all the components were operating exactly as intended, but the complexity of the component interactions led to unanticipated system behavior. These unidentified scenarios included things like missing cases in software requirements and subtle timing errors in communication (sending and receiving messages) between the system components. STPA also identified component failures in the system that could cause hazards. Most traditional hazard analysis methods consider only these types of component failure events.

The Japanese Aerospace Exploration Agency (JAXA) used STPA experimentally on their unmanned spacecraft, called the HTV, which delivers cargo to the International Space Station. STPA found everything identified in the HTV fault tree analysis (required by NASA) plus it found additional hazardous scenarios, mostly related to system design flaws and to software but also related to hazardous interactions among the multiple HTV control agents (astronauts, HTV software, NASA mission controllers and JAXA mission controllers) [Ishimatsu 2013].

Experimental application of STPA to the NextGen In-Trail Procedure (ITP) in a recent MIT research project identified more scenarios than the fault tree and event tree mixture used in the official ITP safety analysis and documented in DO-312 [Fleming, 2013]. STPA identified all the hazard causes produced by the official analysis, but found many more that had been omitted. Using STPA, more safety-related requirements for ITP were generated than are listed in the official RTCA requirements document [8]. The official fault tree/event tree analysis produced a probabilistic risk number for an accident—which was almost certainly not accurate as it omitted many cases in the analysis—while STPA instead identified the potential safety weaknesses in the system so they could be fixed or mitigated.

EPRI (Electric Power Research Institute) ran a comparative evaluation of fault trees, event trees, HAZOP, FMEA, and a few other traditional techniques as well as STPA on a real nuclear power plant design. Each hazard analysis technique was applied by experts on the techniques. STPA was the only one that found a scenario for a real accident that had occurred on that plant design (which, of course, the analysts did not know about).

The original FMEA for a blood gas analyzer (a medical device) that had been recalled by the FDA because of a serious adverse event took a team of people a year to perform and found 75 hazardous scenarios. It did not find the scenario leading to the recall. STPA performed by one person in two weeks found 175 scenarios including 9 leading to the hazardous behavior involved in the recall [Balgos 2012].

To evaluate usefulness and learnability for subject matter experts and system designers, two one-day workshops have been held to teach the technique in the morning and then have the experts apply it to their own system in the afternoon. In both cases, the engineers, even though they had just learned STPA, identified safety design flaws in the systems they were designing or evaluating that they had not noticed before. One typical comment was “We never realized that [system design feature] was important for safety or could lead to an accident.” In these two informal evaluations, one resulted in a recommendation to adopt STPA (for use on a radiation therapy device) and the other to conduct a larger controlled comparison (for U.S. Air Force mission assurance).

There have been many more successful uses of STPA in most every type of industry. What was most surprising was not just that it found more causes of hazards (which could have been predicted from a theoretical comparison), but that STPA took much less time and fewer resources to perform.

Chapter 5: STPA used for Security

William Young
Adam Williams

Chapter 6: Advanced Topics and Future Extensions

Answers to Exercises

Exercise: What are some other examples of indirect causation?

There are a very large number of general examples that might have been listed. Examples of the types of things that might be listed with respect to safety: Promoting productivity over safety, cutting budgets or personnel, relaxing safety standards.

Exercise: Where is the chain of events found in a fault tree, an event tree, HAZOP, and FMEA?

Fault tree: The chains of events are the leaf nodes of the tree, i.e., the cut sets or sets of events that lead to the hazard at the root of the tree. (The intermediate events between the root and leaf nodes are "pseudo events" that just used in the refinement of the top level hazard into the leaf nodes.)

Event tree: The chain of events is listed at the top of the event tree.

HAZOP: Each deviation for each component in the tree is traced to both possible causes, which are the predecessor events and possible consequences, which are the following events. So although the chain is not identified directly (as in the top of the event tree or in the fault tree cut sets or leaf nodes), it is identified implicitly.

FMEA: As with HAZOP, the cause(s) and possible effects or consequences are identified for each failure considered in the FMEA.

Exercise: What are some systemic causes of accidents in systems with which you are familiar?

Examples of things you might have listed are budget pressures, time and schedule pressures, safety culture, a belief that the cost of safety is reduced productivity, beliefs that taking steps to reduce risk is unmanly, budget cuts by management without specifying how lower management levels are to make decisions on where to cut, the influence of politics on decision making, lack of a company safety policy to convey to employees how they should make safety-critical decisions, poor employee morale, employee/management rapport, effects of the legal system on accident investigation and on the collection and exchange of safety information, employee certification, public sentiment, etc.

Exercise: What are some other examples of emergent properties?

Examples of possible answers: security and any of what are usually called system "qualities" or "ilities"

Exercise: What are some of the safety constraints in the systems in your industry? How are they enforced or controlled?

The answers here will be very system specific.

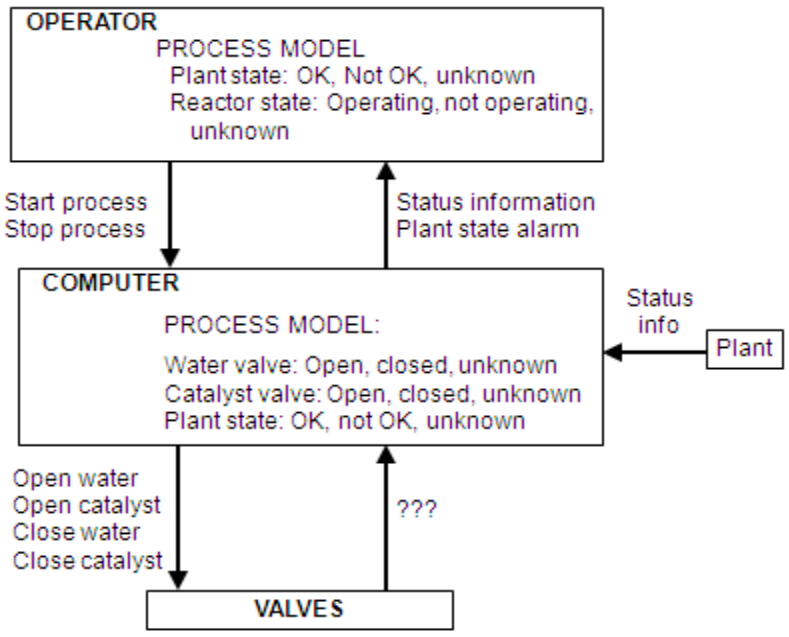
Exercise: For your industry or for a system with which you are familiar, what "process" is being controlled? How is it controlled? What are some typical hazards that must be handled? What are some safety constraints that must be enforced?

Again, the answers will be system-specific.

Exercise: Identify the system-level hazard(s) for the batch reactor.

Pressure or temperature of the chemical reaction exceeds a threshold.

Exercise: Create the functional safety control structure for the batch reactor system shown on page X of ESW. What is an accident in this system? What is the system-level safety constraint involved in the accident?



Exercise: Take the control structure you created for the batch reactor and create the Step 1 tables. Then change the entries into requirements for the operator and the computer software.

Step 1 table for the Software:

Hazard: Catalyst in reactor without reflux condenser operating (water flowing through it)

Control Action	Not providing causes hazard	Providing causes hazard	Too early/too late, wrong order	Stopped too soon/ applied too long
Open water	Not opened when catalyst open		Open water more than X seconds after open catalyst	Stop before fully opened
Close water		Close while catalyst open	Close water before catalyst closes	
Open catalyst		Open when water valve not open	Open catalyst more than X seconds before open water	
Close catalyst	Do not close when water closed		Close catalyst more than X seconds after close water	Stop before fully closed

Safety Constraints:

- Water valve must always be fully open before catalyst valve is opened.
  - Water valve must never be opened (complete opening) more than X seconds after catalyst valve opens
- Catalyst valve must always be fully closed before water valve is closed.
  - Catalyst valve must never be closed more than X seconds after water valve has fully closed.

In the real accident, both of these constraints were missing from the software requirements. Even if the first one had been identified, the second one is the type of case that is often omitted.

Exercise: Take your Step 1 table that you generated for the batch chemical reactor and identify causal scenarios for the unsafe control actions. At the least,

- Identify some causes of the hazardous control action: *Open catalyst valve when water valve not open.* HINT: Consider how controller's process model could identify that the water valve is open when it is not.

  The valve does not open for some physical reason. The design may be such that the software assumes the water valve has opened because an instruction was issued to close it and no feedback was provided in the design to tell the software whether the control action was successfully completed. Or feedback was provided but it says only that the control action signal was received by the valve actuator, but not that the valve actually opened. Valve opening or closing could have failed or been blocked for a variety of reasons but listing these may not be necessary to come up with a solution for the problem.

- What are some causes for a required control action (e.g., open water valve) being given by the software but not executed?

  A lost or corrupted signal, failure of the valve actuator or blocking of the valve before it is fully open or closed, and other failures of the loop components.

- What design features (controls) might you use to protect the system from the scenarios you found?

  One possibility is to use a flow monitor to check that water is actually flowing through the pipe before the catalyst valve is opened and vice versa for the water valve. Other design options are possible. The goal of STPA is not to identify the options for the designers, but to provide information that will help them to make the design decisions.

Exercise: Identify the process model variables for the software controller in the batch reactor system from the previous chapter.

- As the control structure shows, the software is primarily responsible for controlling two valves in the plant. Therefore, at a minimum, the software must have process model variables that reflect the state of those valves. In addition, the software is responsible for stopping operations when something in the plant goes wrong; therefore the software must also know the plant state.

  Software Controller Process Model Variables:
  Water Valve: Open, Closed
  Catalyst Valve: Open, Closed
  Plant State: OK, Not OK

Exercise: Using the process model variables you identified for the batch reactor software, create a context table like Table 3.1 for the Open Catalyst Valve control action.

Control Action	Plant State	Water Valve	Catalyst Valve	Hazardous to provide control action in this context?	Hazardous to provide control action too early in this context?	Hazardous to provide control action too late in this context?
Open Catalyst Valve command	Ok	Closed	Closed	Yes	Yes	Yes
Open Catalyst Valve command	Ok	Closed	Open	Yes	Yes	Yes
Open Catalyst Valve command	Ok	Open	Closed	No	No	No
Open Catalyst Valve command	Ok	Open	Open	No	No	No
Open Catalyst Valve command	Not Ok	Closed	Closed	Yes	Yes	Yes
Open Catalyst Valve command	Not Ok	Closed	Open	Yes	Yes	Yes
Open Catalyst Valve command	Not Ok	Open	Closed	Yes	Yes	Yes
Open Catalyst Valve command	Not Ok	Open	Open	Yes	Yes	Yes

In the first two rows, it is hazardous to command the catalyst valve open because the water valve is closed. Commanding the catalyst valve open with a closed water valve causes a sharp increase in temperature may lead to potential hazards. In the last four rows, it is hazardous to command the catalyst open because the plant is not Ok and any further operation may lead to potential hazards.

Exercise: Using the process model variables you identified for the batch reactor software, create a context table like Table 3.2 for the lack of an Open Catalyst Valve control action.

Control Action	Plant State	Water Valve	Catalyst Valve	Hazardous to NOT provide control action in this context?
Open Catalyst Valve command NOT provided	Ok	Closed	Closed	No
Open Catalyst Valve command NOT provided	Ok	Closed	Open	No
Open Catalyst Valve command NOT provided	Ok	Open	Closed	No
Open Catalyst Valve command NOT provided	Ok	Open	Open	No
Open Catalyst Valve command NOT provided	Not Ok	Closed	Closed	No
Open Catalyst Valve command NOT provided	Not Ok	Closed	Open	No
Open Catalyst Valve command NOT provided	Not Ok	Open	Closed	No
Open Catalyst Valve command NOT provided	Not Ok	Open	Open	No

Although there are situations when an absent open catalyst valve command can adversely affect the operational goals of the plant, there are no situations in which an absent open catalyst valve command will cause a hazard. In other words, it is always safe to not open the catalyst valve. Although this primer focuses on hazard analysis, note that the same approach can also be applied to non-safety-related functional goals of the system too.

References

- E.E. Adams, 1997. Accident causation and the management system, *Professional Safety*, October
- Vincent H. Balgos, 2012. A Systems Theoretic Application to Design for the Safety of Medical Diagnostic Devices, MIT Master's Thesis, February.
- Frank E. Bird and Robert G. Loftus, 1976. *Loss Control Management*, Loganville, GA: The Institute Press

  Nicolas Dulac (2007). A Framework for Dynamic Safety And Risk Management Modeling in Complex Engineering Systems, Ph.D. dissertation, Engineering Systems Division, MIT February 2007.
- FAA, ATO Safety Management System (SMS) Manual.
- Cody H. Fleming, Melissa Spencer, John P. Thomas, Nancy Leveson, and Chris Wilkinson, 2013. Safety assurance in NextGen and complex transportation systems, Safety Science Volume 55, June 2013.
- John Gordon, 1954. Epidemiology in modern Perspective, Proceedings of Royal Society of Medicine, 47(7): 564-570, July.
- L. Michael Hall, 1997. The Non-Aristotelian Systemic Thinking about "Causation" in Complex Systems, http://www.edu365.cat/aulanet/comsoc/visions/documentos/diverses_causation.htm (accessed August 3, 2013)
- H.W. Heinrich, 1931. *Industrial Accident Prevention: A Scientific Approach,* New York: McGraw-Hill
- Takuto Ishimatsu, Nancy G. Leveson, John P. Thomas, Cody H. Fleming, Masafumi Katahira, Yuko Miyamoto, Ryo Ujiie, Haruka Nakao, and Nobuyuki Hoshino, 2013. Hazard Analysis of Complex Spacecraft using STPA, AIAA Journal of Spacecraft and Rockets, in press.
- Alfred Korzybski. (1933/ 1994). Science and sanity: An introduction to non-Aristotelian systems and general semantics, (5th. ed.). Concord, CA: International Society for General Semantics.
- George Lakoff, 2012. Hurricane Sandy: Global warming pure and simple. Salon, November 10, http://www.salon.com/2012/10/31/hurricane_sandy_global_warming_pure_and_simple/ (accessed August 3, 2013).
- Peter Lewycky, 1987. Notes toward an understanding of accident causes, Hazard Prevention, pages 6–8, March/April.
- John Stuart Mill, 1843. *A System of Logic, Ratiocinative, and Inductive: Being a Connected View of the Principle of Evidence and Methods of Scientific Inquiry*. London: J.W. Parker
- Steven J. Pereira, Grady Lee, and Jeffrey Howard, 2006. A System-Theoretic Hazard Analysis Methodology for a Non-advocate Safety Assessment of the Ballistic Missile Defense System, *Proceedings of the 2006 AIAA Missile Sciences Conference*, Monterey, California, November.
- Jens Rasmussen, 1997. Risk management in a dynamic society: A modelling problem, Safety Science, 27 (2-3), pp. 183-213

- Peter M. Senge, 1990. The fifth discipline: The art & practice of the learning organization. NY: Doubleday.
- John Thomas, 2013. Extending and Automating a Systems-Theoretic Hazard Analysis for Requirements Generation and Analysis, Ph.D. Dissertation, MIT Engineering Systems Division.
- Alton L. Thygerson, 1977. *Accidents and Disasters: Causes and Countermeasures*. Englewood Cliffs, New Jersey, Prentice-Hall