STPA HANDBOOK

NANCY G. LEVESON
JOHN P. THOMAS

MARCH 2018

This handbook is intended for those interested in using STPA on real systems. It is not meant to introduce the theoretical foundation, which is described elsewhere. Here our goal is to provide direction for those starting out with STPA on a real project or to supplement other materials in a class teaching STPA.

TABLE OF CONTENTS

Acknowledgements: Many people provided comments on early drafts of this handbook. We want to thank them for their help, including (alphabetically) Matthew Boesch (Ford), Diogo Silva Castilho (MIT and Brazilian Air Force), Christopher Degni (Akamai), Mikela Chatzimichailidou (Imperial College), Rashmi Hegde (Ford), Stephen Johnson (Fluor), Ioana Koglbauer (University of Graz), Galvani Lacerda (Embraer), Simon Lucchini (Fluor), Shem Malquist (FedEx), Maj. Dan Montes (U.S. Air Force), Sandro Nuesch (Ford), Felipe Oliveira (Embraer), Todd Pawlicki (U.C. San Diego Medical Center), Kep Peterson (Akamai), Martin Rezjek (Zurich University of Applied Sciences), Lori Smith (Boeing), Michael Stone (Akamai), Maj. Sarah Summers (U.S. Air Force), Mark Vernacchia (General Motors), Sarra Yako (Ford), Col. William Young (U.S. Air Force), and members of the U.K. National Cyber Security Centre.

Preface

This handbook is intended to assist those who want to start using STPA or who want to try using it for more than simple hazard analysis. As such, we tried to provide examples and have included more examples in the appendices. An appendix is also provided to explain some basic engineering concepts needed to understand and use the handbook to those who are not trained in engineering. Other new concepts are introduced in the main chapters of the handbook itself.

The introduction provides a brief introduction to STAMP, the accident causality model underlying STPA. It also shows examples accidents and explains why STPA is needed for today's complex, software-intensive systems.

The next chapter is an in-depth tutorial on how to perform STPA. It will be useful for the beginner as well as for those who have tried STPA and want to improve their results.

The rest of the handbook describes uses for STPA, including how to integrate it into a standard system engineering process, its use in workplace safety, using STPA for organizational analysis and emergent system properties other than safety, using STPA to provide leading indicators of increasing risk, designing an effective safety management system, and cyber security.

The final chapter describes what we have learned about integrating STPA into a large organization and how to structure the STPA process to make it most effective but also least disruptive to the enterprise.

Our goal in writing this handbook is to provide a guide for those who are actually using STPA rather than writing an academic primer. Therefore, we have omitted references to other work, etc. There are many other sources that provide this type of information but few that focus on instruction and ways to use STPA. To find many examples, published papers, theses, etc., see http://psas.scripts.mit.edu/home/

We encourage users of the handbook to work on examples relevant to their industry as they go through the handbook.

Chapter 1: Introduction

Nancy Leveson

STPA (System-Theoretic Process Analysis) is a relatively new hazard analysis technique based on an extended model of accident causation. In addition to component failures, STPA assumes that accidents can also be caused by unsafe interactions of system components, none of which may have failed. Some of the advantages of STPA over traditional hazard/risk analysis techniques are that:

- Very complex systems can be analyzed. "Unknown unknowns" that were previously only found in operations can be identified early in the development process and either eliminated or mitigated. Both intended and unintended functionality are handled.
- Unlike the traditional hazard analysis methods, STPA can be started in early concept analysis to assist in identifying safety requirements and constraints. These can then be used to design safety (and security) into the system architecture and design, eliminating the costly rework involved when design flaws are identified late in development or during operations. As the design is refined and more detailed design decisions are made, the STPA analysis is also refined to help make more and more detailed design decisions. Complete traceability from requirements to all system artifacts can be easily maintained, enhancing system maintainability and evolution.
- STPA includes software and human operators in the analysis, ensuring that the hazard analysis includes all potential causal factors in losses.
- STPA provides documentation of system functionality that is often missing or difficult to find in large, complex systems.
- STPA can be easily integrated into your system engineering process and into model-based system engineering.

Many evaluations and comparisons of STPA to more traditional hazard analysis methods, such as fault tree analysis (FTA), failure modes and effects criticality analysis (FMECA), event tree analysis (ETA), and hazard and operability analysis (HAZOP) have been done.[1] In all of these evaluations, STPA found all the causal scenarios found by the more traditional analyses but it also identified many more, often software-related and non-failure, scenarios that the traditional methods did not find. In some cases, where there had been an accident that the analysts had not been told about, only STPA found the cause of the accident. In addition, STPA turned out to be much less costly in terms of time and resources than the traditional methods.

Theoretical Foundation

To use STPA requires little understanding of the underlying theoretical, mathematical foundation, but an intuitive understanding is helpful. If this topic does not interest you, we suggest you skip to the next section.

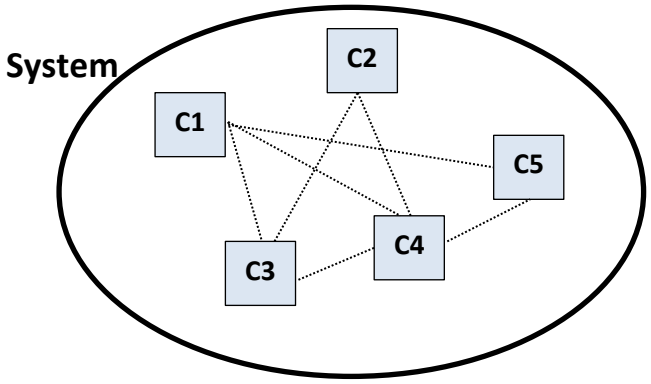
STPA is based on System Theory. Systems theory was developed after World War II to cope with the increasingly complex systems with advanced technology that were being created. First, it is important to understand what it was replacing.

[1] Information about some of these were presented at past MIT STAMP/STPA workshops. Presentations can be found at http://psas.scripts.mit.edu/home/

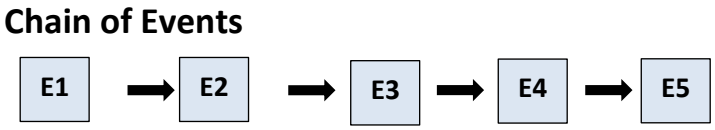
Analytical Decomposition (the traditional approach)

For centuries complexity has been handled by breaking the system into smaller components, examining and analyzing each component in isolation, and then combining the results in order to understand the behavior of the composed components.

The physical or functional components are broken into distinct components that are assumed to interact in direct and known ways. The functional components of an aircraft, for example, might be propulsion, navigation, attitude control, braking, cabin environment control, and so on. Aircraft may also be decomposed into physical components, such as fuselage, engines, wings, stabilizers, nozzles, and so on. Notice that mappings may be made between the functional components and the physical ones but the mappings are assumed to be direct and known.



Behavior, in contrast, is traditionally modeled as separate events over time, where each event is the direct result of the preceding event(s). There may be multiple preceding or following events combined through the use of AND/OR logic but that simply results in multiple chains (with the AND/OR logic used to reduce the number of separate chains needing to be specified by putting them together into trees).



The events may be grouped into operational stages such as for aircraft, pushback, taxi, take off, climb, cruise, approach, and touchdown.

Once the system has been broken into components (pieces), the components are analyzed separately and the results of the separate analyses are combined to create a system analysis. For example, if the weight of an complex object is the analysis goal, the separate pieces may be weighed and the result combined to get the system weight. As another common example, the system reliability is usually evaluated by evaluating the reliability of the individual components and then the component reliabilities are combined mathematically to evaluate the system reliability.

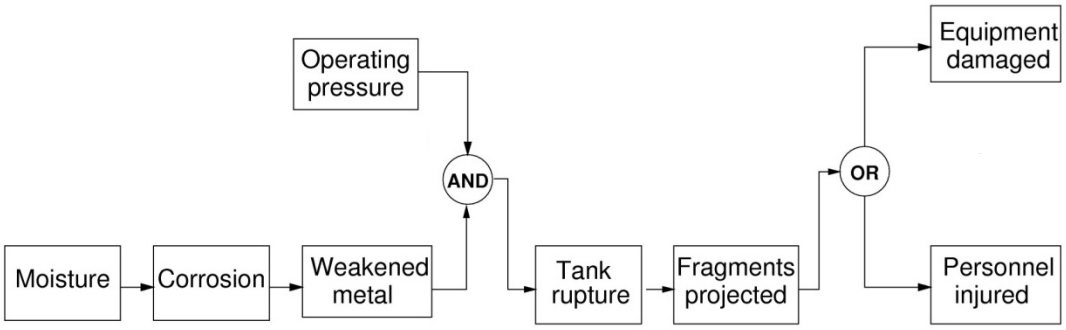
The success of this type of decompositional or reductionist approach relies on the assumption that the separation and individual analysis does not distort the phenomenon or property of interest. More specifically, the approach works if:

- Each component or subsystem operates independently. If events are modeled, then they are independent except for the immediately preceding and following events.
- Components act the same when examined separately (singly) as when they are playing their part in the whole.
- Components and events are not subject to feedback loops and other indirect interactions.
- The interactions among the components or events can be examined pairwise and combined into a composite value.

If these assumptions are not true, then the simple composition of the separate analyses will not accurately reflect the system value.

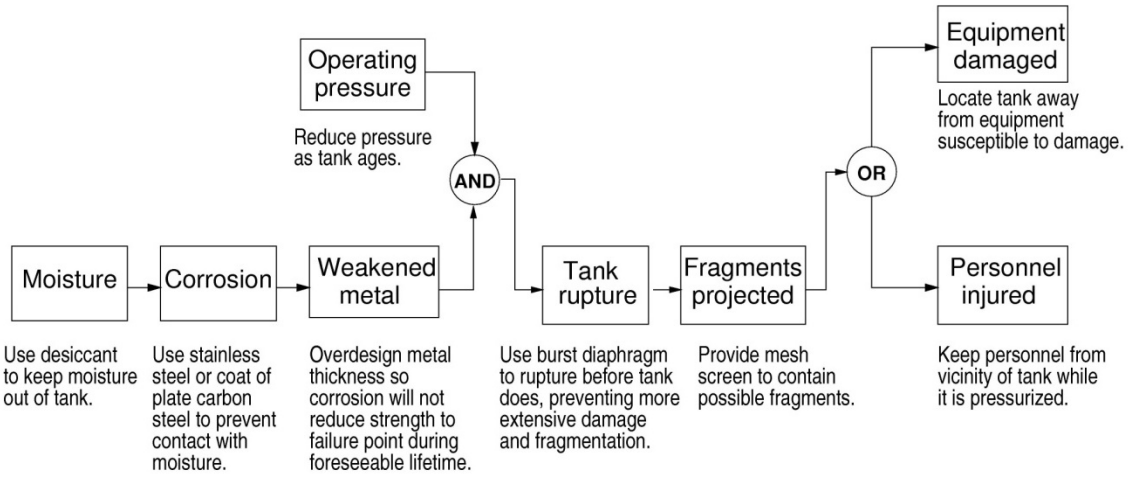
Why is all of this important? Because traditional hazard analysis is based on decomposition and therefore on these assumptions. The basic approach involved is to divide the system into *components*, assume that accidents are caused by component failure, calculate the probability of failure of each component separately, and later combine the analysis results (based on assumptions about the types of interactions among components that can occur) into a system reliability figure, which is assumed to be a measure of safety or risk. Examples are Failure Modes and Effects Analysis (FMEA) and Failure Modes and Effects Criticality Analysis. The latter considers only failures that can lead to a critical loss rather than all failures.

Alternatively, chains of directly related physical or logical (functional) failure *events* that can lead to a loss are identified and the probability of each is combined into a probability of the event chain occurring. The following is a chain of events example for a tank explosion.



In this example, moisture gets into the tank, causing corrosion. The corrosion leads to weakened metal, which, along with a particular operating pressure, can cause the tank to rupture. As a result, fragments are projected and equipment is damaged or personnel are injured.

If accidents are assumed to be caused by failure events, then it makes sense that the approach to preventing accidents is to eliminate the events or to put barriers between the events so that one failure does not cause another event down the chain. The tank rupture model of events is annotated below with the typical types of accident prevention design techniques used, including redundancy, barriers, high component integrity and overdesign, fail safe design, and, for humans, the use of operational procedures, checklists, and training. These design features are used to reduce the probability of the failure events occurring or of them propagating.



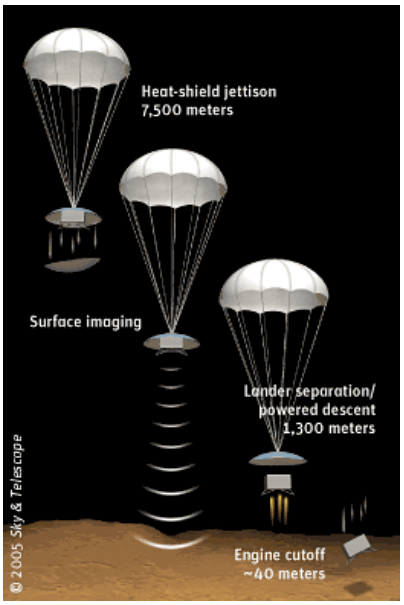
The failure events, of course, must be assumed to be stochastic[2] for probabilities or likelihood to be determined. Unfortunately, software and humans do not satisfy this assumption. This second, event-chain approach to hazard analysis is the basis for fault tree analysis (FTA), event tree analysis (ETA), fault hazard analysis (FHA), and hazard and operability analysis (HAZOP, which uses a deviation rather than a failure as the event or condition to be considered).

The assumptions underlying the decompositional approach to traditional hazard analysis are true (or close enough) for the type of electromechanical systems we used to build, and they are still true for certain properties in our new high-tech, software-intensive systems. However, the greatly increased complexity in our systems today (which is made possible primarily by the use of software) is creating systems where the approach is no longer as effective. It is much more difficult today to anticipate, understand, plan, and guard against all potential system behavior before operational use of our systems. Complexity is creating "unknowns" that cannot be identified by breaking the system behavior into chains of events. In addition, complexity is leading to important system properties (such as safety) not being related to the behavior of individual system components but rather to the interactions among the components. Accidents can occur due to unsafe interactions among components that have not failed and, in fact, satisfy their requirements.

Some examples may be helpful.

Some Navy aircraft were ferrying missiles from one point to another. One pilot executed a planned test by aiming at the aircraft in front (as he had been told to do) and firing a dummy missile. Apparently nobody knew that the "smart" software was designed to substitute a different missile if the one that was commanded to be fired was not in a good position. In this case, there was an antenna between the dummy missile and the target, so the software decided to fire a live missile located in a different (better) position instead. What aircraft component(s) failed here?

[2] Something is stochastic if it can be described by a probability distribution or pattern that may be analyzed statistically to understand average behavior or an expected range of behavior but may not be predicted precisely



This loss involved the Mars Polar Lander. It is necessary to slow the spacecraft down to land safely. Ways to do this include using the Martian atmosphere, a parachute and descent engines (controlled by software). As soon as the spacecraft lands, the software must immediately shut down the descent engines to avoid damage to the spacecraft. Some very sensitive sensors on the landing legs provide this information. But it turned out that noise (false sensor signals) are generated when the legs are deployed. This expected behavior was not in the software requirements. Perhaps it was not included because the software was not supposed to be operating at this time, but the software engineers decided to start early to even out the load on the processor. The software thought the spacecraft had landed and shut down the descent engines while the spacecraft was still 40 meters about the planet surface. Which spacecraft components failed here?

It is dangerous for an aircraft's thrust reversers (which are used to slow the aircraft after it has touched down) to be activated when the aircraft is still in the air. Protection is designed into the software to prevent a human pilot from erroneously activating the thrust reversers when the aircraft is not on the ground. Without going into the details, some of the clues for the software to determine the plane has landed are weight on wheels and wheel spinning rate, which for a variety of reasons did not hold in this case. For example, the runway was very wet and the wheels hydroplaned. As a result, the pilots could not activate the thrust reversers and the aircraft ran off the end of the runway into a small hill. What aircraft components failed here?

This accident involved a Tupelov aircraft landing in Moscow in 2012. A soft touchdown made runway contact a little later than usual. With the crosswind at the time, the weight-on-wheels switches did not activate and the thrust reverse system would not deploy. The pilots thought that the thrust reversers would deploy as they always do. So with limited runway space to stop the aircraft, they quickly engaged high engine power to stop quicker. Instead, this accelerated the aircraft forward, eventually colliding with a highway embankment. What aircraft components failed here?

The coastal area of Washington state has lots of water and islands. Car ferries are necessary to move people around. One day, the ferry system was brought to its knees when rental cars could not be driven off the ferries after arriving in port. It turned out that a local rental car company had installed a security device to prevent theft by disabling cars if the car moved when the engine was stopped. When the ferries moved and the cars were not running, the cars all disabled themselves and the ferry system was brought to a standstill until enough tow trucks could be found to tow the cars off the ferries. What car component(s) failed here?

The manufacturer of the lithium-ion batteries involved in this incident determined that battery cell venting would occur once in 10 million flight hours, but two batteries malfunctioned in just two weeks in 2013. While there were many factors involved, one of the factors makes an interesting example here. In the event of a battery malfunction, an environmental control system was designed to remove smoke through cooling duct fans by actuating certain valves. However, the unit providing power to the environmental control system simultaneously shut down due to the battery malfunction. As a result, the valves could not be actuated and smoke being generated by the APU battery could not be effectively directed outside the passenger cabin and battery compartment.

A more complex example in aircraft avionics systems can be found in Appendix A. In addition, Dr. Ioana Koglbauer at the University of Graz (Austria) provided an example that emphasizes the usefulness of a systems approach. A certified fixed-base flight simulator at their university has the same production cockpit components (e.g. rudder-pedals) as the certified aircraft, but the pedals in the simulator often break and must be replaced. If you try to understand why and solve the problem at the component level, you could blame and punish the pilot who produced the damage, train the pilot on how to handle the simulator, or invest in improving the pedals. If you look at interactions among components (people, hardware, software), and try to understand why the pilots use the pedals differently in the simulator than in a real aircraft , you come to a different problem definition and solution. It turns out that when the pilots brake in the simulator, the simulator software is not programmed to show the typical small pitch down movement when the aircraft comes to a stop. This movement can also be felt when parking a car. In the absence of this feedback in the aircraft simulator, the pilots apply the brakes longer and stronger than needed. For this reason, the pedals break in the simulator, but not in the real aircraft. The best solution is to improve the software and give pilots the feedback they need.

In all of these cases (and in hundreds of others), the problems did not stem from individual component failures but from flaws in the system engineering process that resulted in flawed system designs. Decompositional analysis cannot identify these causes of accidents including human error, software requirements errors, and system design flaws. We need something different. That different theoretical foundation, which underlies STPA, is called System Theory.

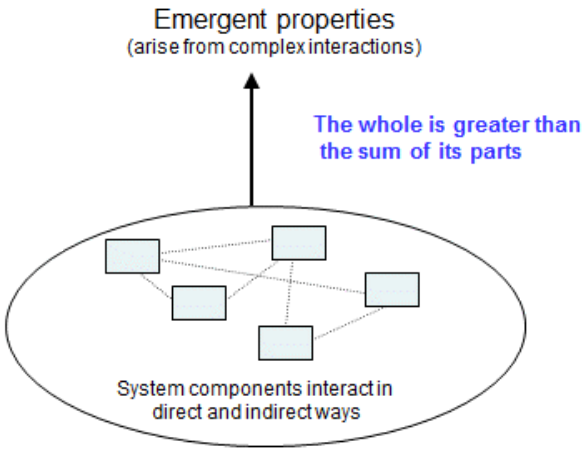
System Theory

As stated above, system theory as used in engineering was created after World War II to deal with the increased complexity of the systems being built after the war.[3] It was also created for biology in order to successfully understand the complexity of biological systems. In these systems, separation and analysis of separate, interacting components (subsystems) distorts the results for the system as a whole because the component behaviors are coupled in non-obvious ways. The first engineering uses of these new ideas were in the missile and early warning systems of the 1950s and 1960s.

Some unique aspects of System Theory are:

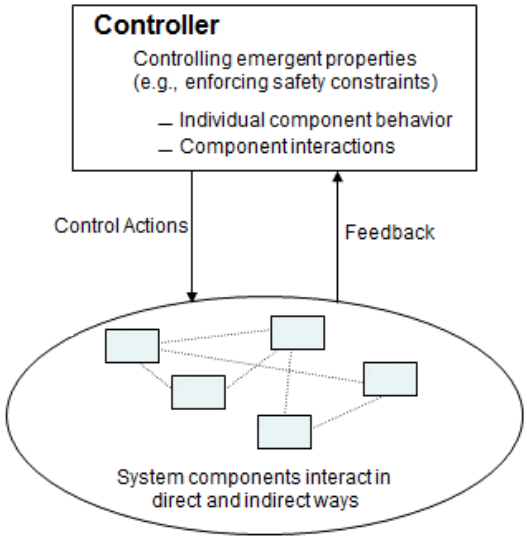
- The system is treated as a whole, not as the sum of its parts. You have probably heard the common statement: “the whole is more than the sum of its parts.”
- A primary concern is *emergent properties*, which are properties that are not in the summation of the individual components but “emerge” when the components interact. Emergent properties can only be treated adequately by taking into account all their technical and social aspects.
- Emergent properties arise from relationships among the parts of the system, that is, by how they interact and fit together.

[3] If you are interested in learning more, we recommend Peter Checkland, *System Thinking, System Practice*, John Wiley & Sons (1981), Gerald Weinberg, *An Introduction to General Systems Thinking*, John Wiley & Sons (1975), and perhaps for historical interest, W. Ross Ashby, *An Introduction to Cybernetics*, Chapman and Hall (1956).



If emergent properties arise from individual component behavior and from the interactions among components, then it makes sense that controlling emergent properties, such as safety, security, maintainability, and operability, requires controlling the behavior of the individual components and the interactions among the components. We can add a controller to the figure. The controller provides control actions on the system and gets feedback to determine the impact of the control actions. If this looks like a standard feedback control loop to you, you are absolutely correct. That is what it is.

The controller enforces constraints on the behavior of the system. Example safety constraints might be that aircraft or automobiles must remain a minimum distance apart, pressure in deep water wells must be kept below a safe level, aircraft must maintain sufficient lift to remain airborne unless landing, accidental detonation of weapons must be prevented, and toxic substances must never be released from a plant.



To explain the model above using a simple example, consider the national or international airspace. If each airline were allowed to optimize its schedule by flying at any time and using any route, chaos might result if everyone tried to land at Chicago, New York, or Heathrow at 5 pm. To avoid such conflicts, system-level air traffic control (ATC) is introduced that controls two emergent properties: throughput and safety.

ATC is responsible for optimizing the overall throughput in the system (which may not optimize the route for every aircraft) and for maintaining adequate separation among the aircraft.

This model includes everything we now do in safety engineering. Control is interpreted broadly. For example, component failures and unsafe interactions may be controlled through design, such as using redundancy, interlocks, barriers, or fail-safe design. Safety may also be controlled through process, such as development processes, manufacturing processes and procedures, maintenance processes, and general system operating processes. Finally, safety may be controlled using social controls including government regulation, culture, insurance, law and the courts, or individual self-interest. Human behavior can be partially controlled through the design of the societal or organizational incentive structure.

What is STAMP?

STAMP[4] (System-Theoretic Accident Model and Processes) is the name of the new accident causality model based on systems theory,[5] as described in the previous section, which provides the theoretical foundation for STPA. It expands the traditional model of causality beyond a chain of directly-related failure events or component failures to include more complex processes and unsafe interactions among system components, and it underlies STPA and other tools.

In STAMP, safety is treated as a dynamic control problem rather than a failure prevention problem. No causes are omitted from the STAMP model, but more are included and the emphasis changes from preventing failures to enforcing constraints on system behavior.

Some advantages of using STAMP are that:

- It works on very complex systems because it works top-down rather than bottom up.
- It includes software, humans, organizations, safety culture, etc. as causal factors in accidents and other types of losses without having to treat them differently or separately.
- It allows creating more powerful tools, such as STPA, accident analysis (CAST), identification and management of leading indicators of increasing risk, organizational risk analysis, etc.

Because STAMP applies to any emergent property, STPA can be used for any system property, including cybersecurity.

STAMP is not an analysis method. Instead it is a model or set of assumptions about how accidents occur. STAMP is an alternative to the chain-of-failure-events (or dominos or Swiss cheese slices, all of which are essentially equivalent) that underlies the traditional safety analysis techniques (such as Fault Tree Analysis, Event Tree Analysis, HAZOP, FMECA, and HFACS). Just as the traditional analysis methods are constructed on the assumptions about why accidents occur in a chain-of-failure-events model, new analysis methods can be constructed using STAMP as a basis. Note that because the chain-of-failure events model is a subset of STAMP, tools built on STAMP can include as a subset all the results derived using the older safety analysis techniques.

The two most widely used STAMP-based tools today are STPA (System Theoretic Process Analysis) and CAST (Causal Analysis based on Systems Theory). STPA is a proactive analysis method that analyzes the potential cause of accidents during development so that hazards can be eliminated or controlled.

---

[4] Nancy G. Leveson, *Engineering a Safer World*, MIT Press (2012)

[5] The basic system theory that underlies STAMP should not be confused with Complexity Theory, which later grew out of basic system theory to explain the adaptive, non-linear behavior common in nature and social organizations. For engineering, even the engineering or design of organizations, basic system theory suffices. For more, see Appendix F.

CAST is a retroactive analysis method that examines an accident/incident that has occurred and identifies the causal factors that were involved. This handbook concentrates on the use of STPA. A future, similar handbook is planned for CAST.

Organization of this Handbook

Chapter 2 of this handbook explains the basics of STPA and how to do it. An aerospace example is used, but examples from other industries are included in the appendices.

Chapters 3 through 7 describe how to use STPA in various common types of engineering activities. Because STPA can be applied to any emergent (system) property, Chapter 5 applies STPA to something other than safety, in this case, system engineering effectiveness. While most of the applications of STPA in industry so far have involved safety, companies are now starting to use it for other system properties such as quality, security, and production engineering.

Finally, Chapter 8 provides some advice about how to integrate STPA into a large organization. The appendices provide additional examples, a guide for analyzing an organizational safety management system, some additional explanation (including an example) about why decomposition does not work for today's software-intensive, complex systems, and an introduction to some basic concepts for non-engineers.

Chapter 2: How to Do a Basic STPA Analysis

John Thomas

STPA Method Overview

The steps in basic STPA are shown in Figure 2.1 along with a graphical representation of these steps.

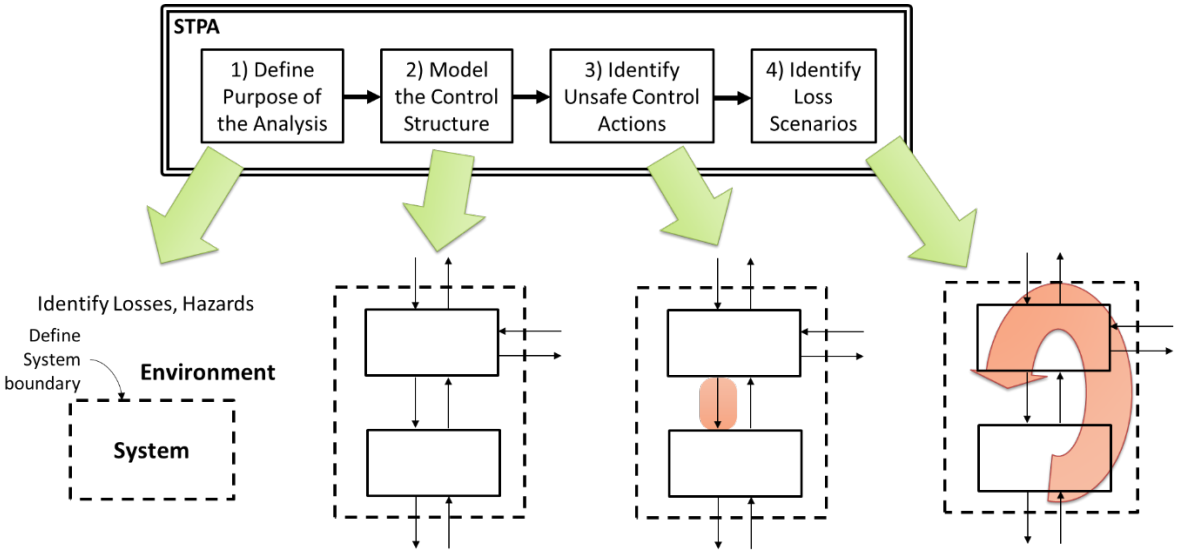


Figure 2.1: Overview of the basic STPA Method

Defining the purpose of the analysis is the first step with any analysis method. What kinds of losses will the analysis aim to prevent? Will STPA be applied only to traditional safety goals like preventing loss of human life or will it be applied more broadly to security, privacy, performance, and other system properties? What is the system to be analyzed and what is the system boundary? These and other fundamental questions are addressed during this step.

The second step is to build a model of the system called a control structure. A control structure captures functional relationships and interactions by modeling the system as a set of feedback control loops. The control structure usually begins at a very abstract level and is iteratively refined to capture more detail about the system. This step does not change regardless of whether STPA is being applied to safety, security, privacy, or other properties.

The third step is to analyze control actions in the control structure to examine how they could lead to the losses defined in the first step. These *unsafe control actions* are used to create functional requirements and constraints for the system. This step also does not change regardless of whether STPA is being applied to safety, security, privacy, or other properties.

The fourth step identifies the reasons why unsafe control might occur in the system. Scenarios are created to explain:

1. How incorrect feedback, inadequate requirements, design errors, component failures, and other factors could cause unsafe control actions and ultimately lead to losses.
2. How safe control actions might be provided but not followed or executed properly, leading to a loss.

Once scenarios are identified, they can be used to create additional requirements, identify mitigations, drive the architecture, make design recommendations and new design decisions (if STPA is used during development), evaluate/revisit existing design decisions and identify gaps (if STPA is used after the design is finished), define test cases and create test plans, develop leading indicators of risk, and for other uses as described in later chapters of this handbook.

Defining the purpose of the analysis

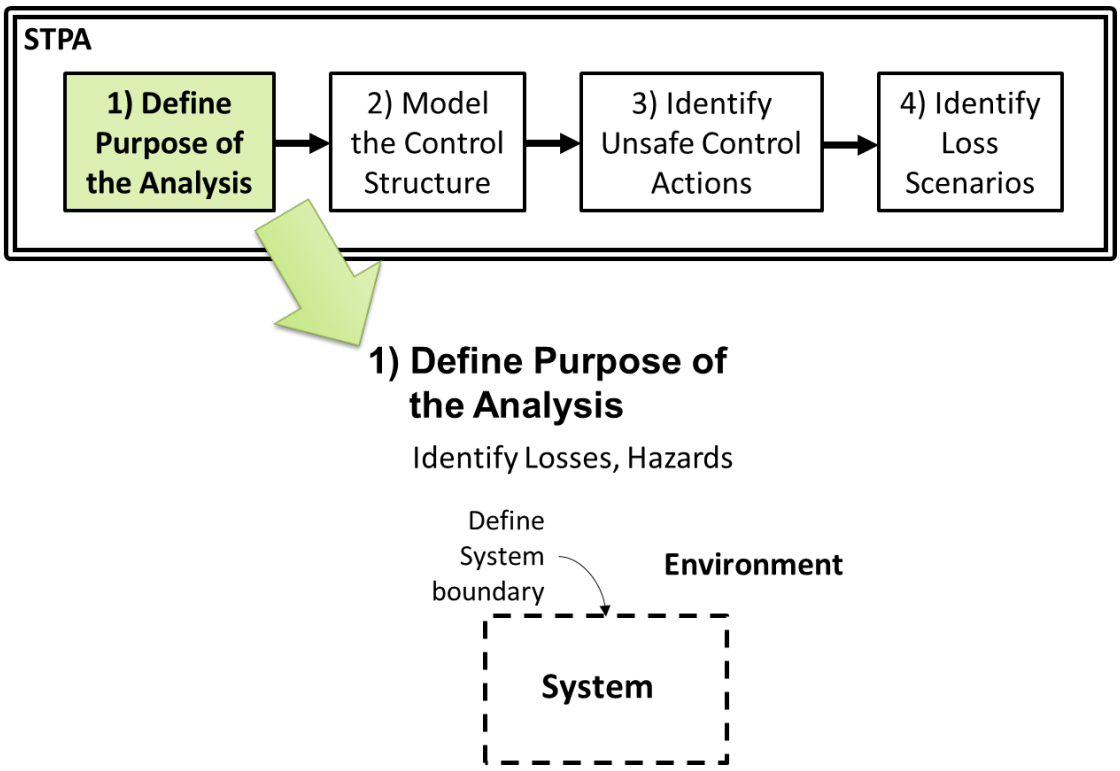


Figure 2.2: Defining the Purpose of the Analysis

As Figure 2.2 shows, the first step in applying STPA is to define the purpose of the analysis. Defining the purpose of the analysis has four parts:

1. Identify losses
2. Identify system-level hazards
3. Identify system-level constraints
4. Refine hazards (optional)

Figure 2.3 summarizes the four parts described in this section.

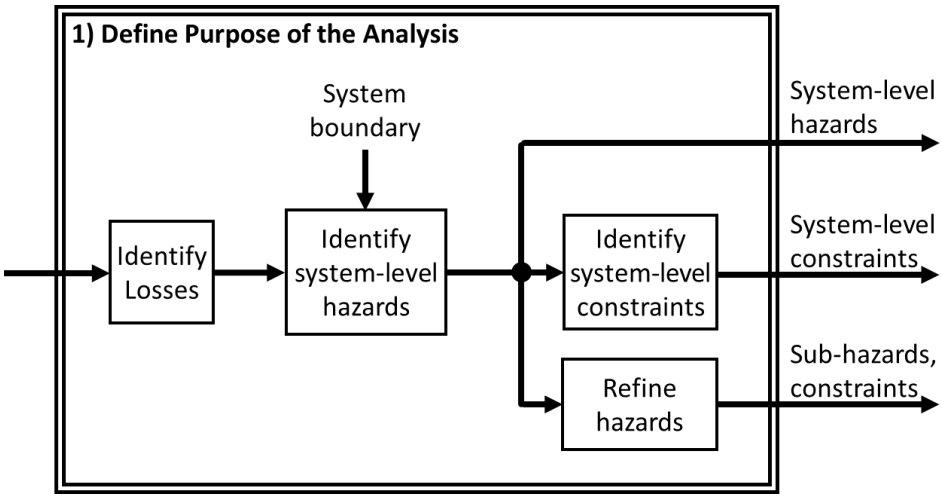


Figure 2.3: Overview of defining the analysis purpose

Identifying losses

> Definition: A *loss* involves something of value to stakeholders. Losses may include a loss of human life or human injury, property damage, environmental pollution, loss of mission, loss of reputation, loss or leak of sensitive information, or any other loss that is unacceptable to the stakeholders.

Different words are used to identify the goal of hazard analysis in different industries, for example, preventing an accident, a mishap, or an adverse event. To eliminate confusion about multiple terms used by practitioners in various industries, "losses" will be the term used in this chapter and the goal of STPA is to prevent losses.

STPA can be used to target any loss that is unacceptable to stakeholders. If more than one loss is included, they can be ranked and prioritized. Because every STPA result will be traceable to one or more losses, the analysis results can be easily ranked and prioritized based on the losses to which they refer.

Before any analysis begins, the stakeholders must identify the losses on which they want the analysis to focus. The losses to be considered may be defined by management, by government regulations, or by customers. A general approach to identifying losses may involve:

1. Identify the stakeholders, e.g. Users, producers, customers, operators, etc.
2. Stakeholders identify their "stake" in the system. What do they value? For example, human life, fleet of useable aircraft, electrical power generation, transportation, etc. What are their goals? For example, to maintain a fleet of useable aircraft, to provide transportation, to provide medical treatment, to provide electrical power generation, etc.
3. Translate each value or goal into a loss, e.g. loss of life, loss of aircraft, loss of electrical power generation, loss of transportation, etc.

The following list provides some examples of losses that users of the analysis often want to avoid:

L-1: Loss of life or injury to people
L-2: Loss of or damage to vehicle
L-3: Loss of or damage to objects outside the vehicle
L-4: Loss of mission (e.g. transportation mission, surveillance mission, scientific mission, defense mission, etc.)
L-5: Loss of customer satisfaction

L-6: Loss of sensitive information
L-7: Environmental loss
L-8: Loss of power generation

Some tips to prevent common mistakes when identifying losses:

1. Losses can include any loss that is unacceptable to any stakeholder
2. Losses should not reference individual components or specific causes like “human error” and “brake failure”
3. Losses may involve aspects of the environment that are not directly controlled by the system designer
4. Document any special considerations or assumptions made, such as losses that are explicitly excluded

After the losses of concern in the analysis are identified, the next step is to define the hazards related to these losses.

Identifying system-level hazards

Definition: A *hazard* is a system state or set of conditions that, together with a particular set of worst-case environmental conditions, will lead to a loss.

Definition: A *system* is a set of components that act together as a whole to achieve some common goal, objective, or end. A system may contain subsystems and may also be part of a larger system.

To identify the system-level hazards, you will first need to identify the system to be analyzed and the system boundary. A system is an abstraction conceived by the analyst. A decision must be made about what is included in the system and what the system boundary is. With respect to engineering, the most useful way to define the system boundary for analysis purposes is to include the parts of the system over which the system designers have some control. This is the primary reason for distinguishing between hazards and losses—losses may involve aspects of the environment over which the system designer or operator has only partial control or no control at all. The goal of safety engineering is to eliminate or mitigate the effects of hazards in the system under analysis so some level of control is necessary. Figure 2.4 illustrates the system boundary as an abstraction that separates a system from its environment.

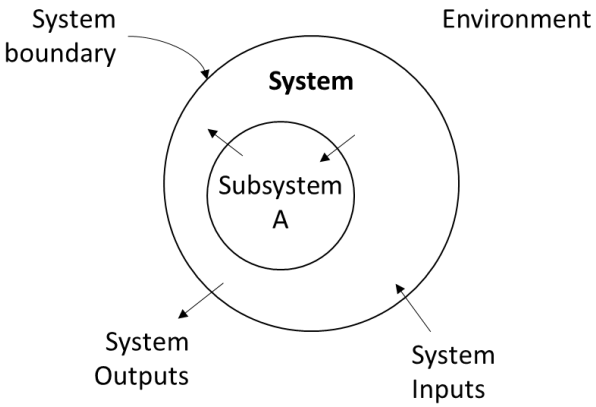


Figure 2.4: The relationship between a system, system boundary, and the environment

As an example, consider a nuclear power plant. A release of radioactive materials, the proximity of a nearby population or city, and the direction of the wind may all be important factors that lead to a potential loss of life. However, as engineers we cannot control the wind and we may not be able to control the city location, but we *can* control the release of radioactive materials in or outside the plant (a system-level hazard).

Once the system and system boundary is identified, the next step is to define the system-level hazards by identifying system states or conditions that will lead to a loss in worst-case environmental conditions. The following list provides some examples of system-level hazards:

H-1: Aircraft violate minimum separation standards in flight [L-1, L-2, L-4, L-5]
H-2: Aircraft airframe integrity is lost [L-1, L-2, L-4, L-5]
H-3: Aircraft leaves designated taxiway, runway, or apron on ground [L-1, L-2, L-5]
H-4: Aircraft comes too close to other objects on the ground [L-1, L-2, L-5]
H-5: Satellite is unable to collect scientific data [L-4]
H-6: Vehicle does not maintain safe distance from terrain and other obstacles [L-1, L-2, L-3, L-4]
H-7: UAV does not complete surveillance mission [L-4]
H-8: Nuclear power plant releases dangerous materials [L-1, L-4, L-7, L-8]

In general, a hazard can lead to one or more losses and each hazard should be traced to the resulting losses. This traceability is typically documented in brackets after the hazard description. The examples above show the traceability to the examples of losses on the previous page.

There are three basic criteria for defining system-level hazards:

- Hazards are *system* states or conditions (not component-level causes or environmental states)
- Hazards *will* lead to a loss in some worst-case environment
- Hazards must describe states or conditions to be prevented

First, the system-level hazards are system states or conditions, not external environmental states that are outside the designer's control. In addition, system-level hazards should not describe detailed component-level causes like a physical component failure (e.g. a hydraulic leak, insufficient brake fluid, etc.). Referencing component-level causes during this step will overly restrict the analysis, making it easy to overlook other less obvious causes during later steps. Instead, identify the system-level states or conditions to be prevented (the hazards) and allow the later STPA steps to systematically identify component-level causes of the hazards.

Second, there must be a worst-case environment in which hazards will lead to a loss. This requirement does not necessarily guarantee that a hazard will *always* result in a loss. For example, a chemical plant may release toxic materials but wind and weather conditions prevent the toxic materials from impacting nearby personnel and populated regions. However, in a worst-case environment, the toxic materials can be carried to populated areas and lead to losses.

Finally, hazards are states or conditions to be prevented. "Aircraft in flight" is a system state that could arguably lead to a loss in a worst-case environment, but it is not a condition to be eliminated or prevented (otherwise we would not be building aircraft). Hazards should be states to be prevented and states that we never want the system to get into—not states that the system must normally be in to accomplish its goals.

Common mistakes when identifying system-level hazards

Confusing hazards with causes of hazards

A common mistake in defining hazards is to confuse hazards with causes of hazards. For example, "brake failure", "brake failure not annunciated", "operator is distracted", "engine failure", and "hydraulic leak" are not system-level hazards but potential *causes* of hazards. To avoid this mistake, make sure the identified hazards do not refer to individual components of the system, like brakes, engines, hydraulic lines, etc. Instead, the hazards should refer to the overall system and system states. In other words, check that each hazard contains:

<Hazard specification> = <System> & <Unsafe Condition> & <Link to Losses>

E.g. H-1 = Aircraft violate minimum separation standards in flight [L-1, L-2, L-4, L-5]

The exact sequence is not important—you could just as easily write "Minimum separation standards for aircraft are violated [L-1, L-2, L-4, L-5]". What is important is that the system-level hazards contain these elements.

Too many hazards containing unnecessary detail

Like losses, there are no hard limits on the number of system-level hazards to include. As a rule of thumb, if you have more than about seven to ten system-level hazards, consider grouping or combining hazards to create a more manageable list. You may be including unnecessary detail and making the list unmanageable, difficult to review, and harder to identify things that are missing. Instead, begin with a more abstract and manageable set of system-level hazards and refine them into sub-hazards later if needed (as explained in the section on refining hazards below).

Ambiguous or recursive wording

The system-level hazards define exactly what "unsafe" means at the system level. A common mistake is to use the word "unsafe" in the hazards themselves. Doing so creates a recursive definition and does not add information or value to the analysis. For example, it might be tempting to write "H-1: Aircraft experiences unsafe flight [L-1]". It can be tempting because it certainly sounds dangerous—an unsafe flight by definition must be hazardous, right? The problem is that it is too vague and does not help specify the actual condition that is unsafe. Some have fallen into the same trap with statements like "H-1: Aircraft experiences an unsafe state [L-1]", only to struggle with the rest of the analysis and miss important cases. A simple solution is to avoid using the word "unsafe" in the hazard itself and instead specify exactly what is meant by "unsafe"—what system states or conditions would make it unsafe? For example, an aircraft that is uncontrolled or that is too close to other aircraft would be unsafe. As you will see, specifying actual conditions like this is extremely useful during later STPA steps.

Confusing hazards with failures

Professionals who are experienced in other hazard analysis methods sometimes fall into the trap of writing STPA hazards describing potential deviations from specified technical functions or describing physical component failures. You may be familiar with traditional techniques that begin by searching for a set of deviations, faults, or functional failures in the technical system. To identify a broader set of causes in STPA, we cannot assume that the defined and specified functions are safe and correct, that human operators will perform as expected, that automated behaviors will not induce human error or confusion, that off-nominal cases will not occur, or that the technical design, specification, and requirements are correct. For example, the hazard "Controlled flight of aircraft into terrain" can be included in STPA while it may be omitted by efforts to examine only purely technical functional failures.

Hazard identification in STPA is about system states and conditions that are inherently unsafe—regardless of the cause. In fact, the system hazards should be specified at a high-enough level that does not distinguish between causes related to technical failures, design errors, flawed requirements, or human procedures and interactions.

What should I look for when reviewing hazards?

Tips to prevent common mistakes when identifying hazards

- Hazards should not refer to individual components of the system
- All hazards should refer to the overall system and system state
- Hazards should refer to factors that can be controlled or managed by the system designers and operators
- All hazards should describe system-level conditions to be prevented
- The number of hazards should be relatively small, usually no more than 7 to 10
- Hazards should not include ambiguous or recursive words like "unsafe", "unintended", "accidental", etc.

Note that STPA is an iterative method and the hazards need not be set in stone at this point. Later STPA steps may uncover new hazards and this list can be revisited and revised as needed.

Defining system-level constraints

Definition: A system-level constraint specifies system conditions or behaviors that need to be satisfied to prevent hazards (and ultimately prevent losses)

Once the system-level hazards are identified, it is straightforward to identify system-level constraints that must be enforced: simply invert the condition.

<Hazard> = <System> & <Unsafe Condition> & <Link to Losses>
<System-level Constraint> = <System> & <Condition to Enforce> & <Link to Hazards>

H-1: Aircraft violate minimum separation standards [L-1, L-2, L-4, L-5]
SC-1: Aircraft must satisfy minimum separation standards from other aircraft and objects [H-1]

H-2: Aircraft airframe integrity is lost [L-1, L-2, L-4, L-5]
SC-2: Aircraft airframe integrity must be maintained under worst-case conditions [H-2]

Each constraint can be traceable to one or more hazards, and each hazard is traceable to one or more losses. In general, the traceability need not be one-to-one; a single constraint might be used to prevent more than one hazard, multiple constraints may be related to a single hazard, and each hazard could lead to one or more losses.

Constraints can also define how the system must minimize losses in case the hazards do occur. For example, if aircraft do violate minimum separation then the violation must be detected and measures must be taken to keep the aircraft from colliding. If a chemical plant does release toxic chemicals, then the toxic environment must be detected and appropriate measures taken. These constraints can generally be written as:

<System-level Constraint> = If <hazard> occurs, then <what needs to be done to prevent or minimize a loss> & <Link to Hazards>

SC-3: If aircraft violate minimum separation, then the violation must be detected and measures taken to prevent collision [H-1]

The system-level constraints should not specify a particular solution or implementation. For example, instead of specifying solutions like installing a collision avoidance system, SC-3 simply states that the

violation must be detected and there must be some way to prevent collision. Specifying a particular solution is usually premature at this early stage and can result in alternative and potentially better solutions being overlooked.

The rest of the STPA analysis will systematically identify scenarios that can violate these constraints, leading to system-level hazards and losses.

Refining the system-level hazards (optional)

Once the list of system-level hazards has been identified and reviewed, these hazards can be refined into sub-hazards if appropriate. Sub-hazards are not necessary for many STPA applications[6], but they can be useful for large analysis efforts and complex applications because they can guide future steps like modeling the control structure (see the next section about control structures).

The first step in refining the system-level hazards is to identify basic system processes or activities that need to be controlled to prevent system hazards. For example, consider the system-level hazard we identified earlier for commercial aviation:

H-4: Aircraft comes too close to other objects on the ground [L-1, L-2, L-5]

One way to derive sub-hazards is to ask: What do we need to control to prevent this hazard? To control the aircraft on the ground, we will need some way to control aircraft deceleration, acceleration, and steering. If these are not controlled adequately (for example if the deceleration is insufficient), it could lead to a system-level hazard.

The following sub-hazards can be derived for H-4:

H-4: Aircraft comes too close to other objects on the ground [L-1, L-2, L-5]

Deceleration
H-4.1: Deceleration is insufficient upon landing, rejected takeoff, or during taxiing
H-4.2: Asymmetric deceleration maneuvers aircraft toward other objects
H-4.3: Deceleration occurs after V1 point during takeoff
Acceleration
H-4.4: Excessive acceleration provided while taxiing
H-4.5: Asymmetric acceleration maneuvers aircraft toward other objects
H-4.6: Acceleration is insufficient during takeoff
H-4.7: Acceleration is provided during landing or when parked
H-4.8: Acceleration continues to be applied during rejected takeoff
Steering
H-4.9: Insufficient steering to turn along taxiway, runway, or apron path
H-4.10: Steering maneuvers aircraft off the taxiway, runway, or apron path

Each of these sub-hazards can be used to produce more specific constraints. *Table 2.1* shows the constraints derived for deceleration based on the sub-hazards above.

Table 2.1: Deriving specific process constraints from the sub-hazards

Sub-hazards derived from H-4	Example constraints
H-4.1: Deceleration is insufficient upon landing, rejected takeoff, or during taxiing	SC-6.1: Deceleration must occur within TBD seconds of landing or rejected takeoff at a rate of at least TBD m/s^2

[6] Those new to STPA may find it helpful to begin with smaller applications that do not require this step.

H-4.2: Asymmetric deceleration maneuvers aircraft toward other objects	SC-6.2: Asymmetric deceleration must not lead to loss of directional control or cause aircraft to depart taxiway, runway, or apron
H-4.3: Deceleration occurs after V1 point during takeoff	SC-6.3: Deceleration must not be provided after V1 point during takeoff

Modeling the control structure

The next step in STPA is to model the hierarchical control structure, as Figure 2.5 shows.

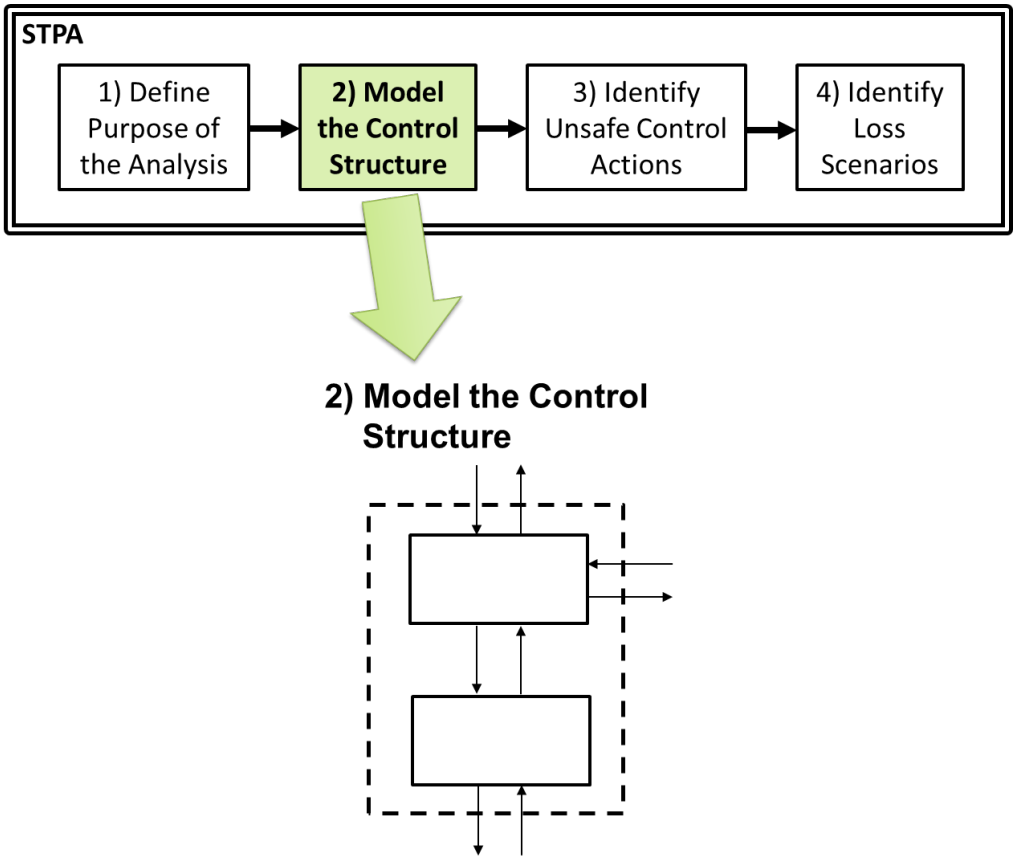


Figure 2.5: Modeling the control structure

What is a control structure?

> Definition: A hierarchical control structure is a system model that is composed of feedback control loops. An effective control structure will enforce constraints on the behavior of the overall system.

A hierarchical control structure is composed of control loops like the one shown in Figure 2.6. In general, a controller may provide control actions to control some process and to enforce constraints on the behavior of the controlled process. The control algorithm represents the controller's decision-making process—it determines the control actions to provide. Controllers also have process models that represent the controller's internal beliefs used to make decisions. Process models may include beliefs

about the process being controlled or other relevant aspects of the system or the environment. Process models may be updated in part by feedback used to observe the controlled process.

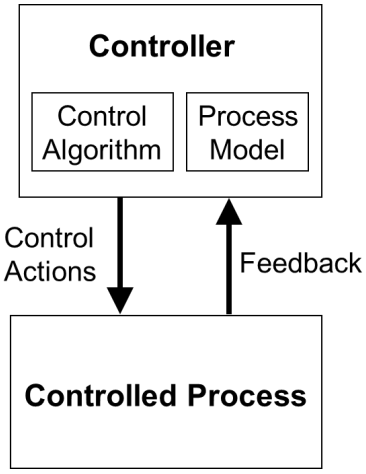


Figure 2.6: Generic control loop

Problems can occur at any point in Figure 2.6. For example, a process model that is not consistent with reality (e.g. a controller believes an aircraft is descending when it is really ascending or believes a car is in Park when it is really in Reverse, the airport runway is empty, etc.) can lead to control actions that are unsafe. A sensor failure may cause incorrect feedback and lead to unsafe behavior. A design may be missing the necessary feedback or may provide delayed feedback that results in a process model flaw and unsafe behavior. As you'll see, STPA provides a way to systematically identify these and other scenarios that can lead to a loss.

The generic control loop in Figure 2.6 can be used to explain and anticipate complex software and human interactions that can lead to losses—two of the biggest challenges in modern engineering. For humans, the process model is usually called a mental model and the control algorithms may be called operating procedures or decision-making rules, but the basic concept is the same.

Of course, most systems typically have several overlapping and interacting control loops. Multiple interacting control loops can be modeled in a hierarchical control structure, as shown in Figure 2.7. A simple example for aviation is shown in Figure 2.8.

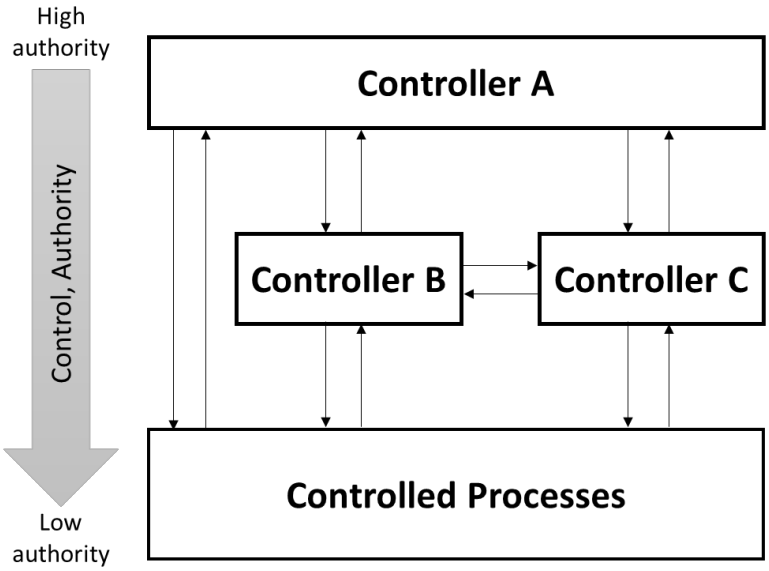


Figure 2.7: Generic hierarchical control structure

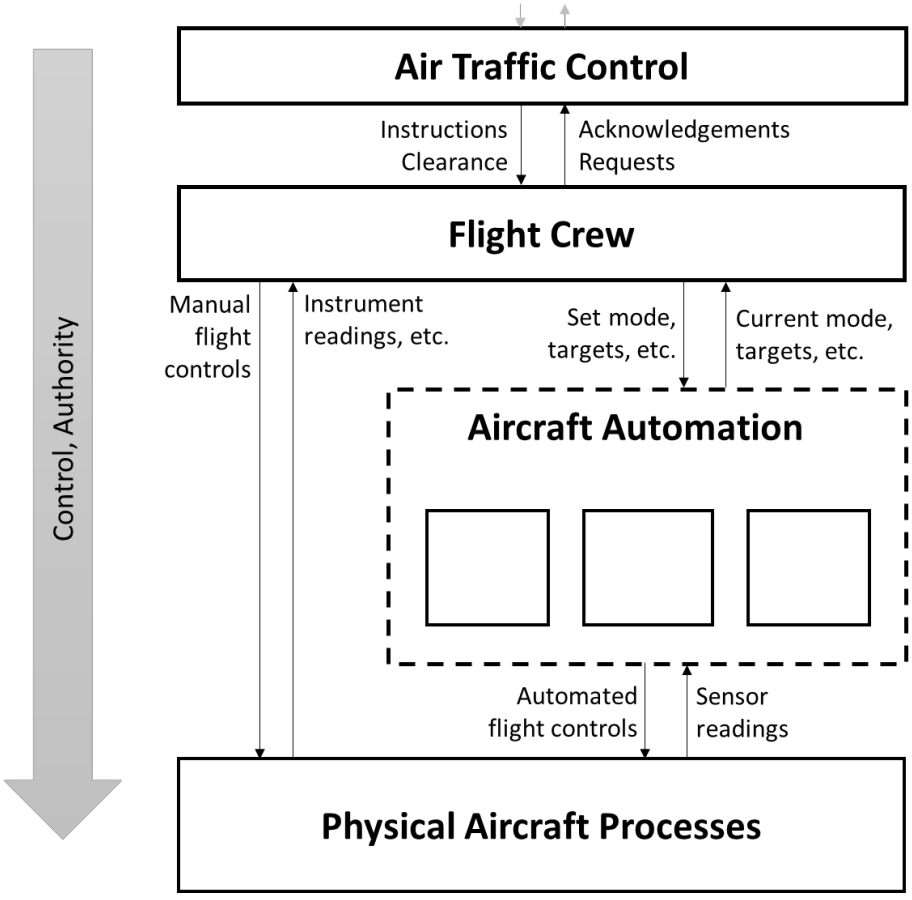


Figure 2.8: Simple example of a hierarchical control structure for aviation

In general, a hierarchical control structure contains at least five types of elements[7]:

- Controllers
- Control Actions
- Feedback
- Other inputs to and outputs from components (neither control nor feedback)
- Controlled processes

The vertical axis in a hierarchical control structure is meaningful: it indicates control and authority within the system. The vertical placement represents the hierarchy of control from high-level controllers at the top to the lowest-level entities at the bottom. Each entity has control and authority over the entities immediately below it, and each entity is likewise subject to control and authority from the entities immediately above. For example, the aircraft automation in Figure 2.8 can act as a controller by sending control actions to physical aircraft systems and monitoring feedback. At the same time, the aircraft automation is also a controlled process that receives and executes control actions from the flight crew and sends feedback to the crew.

In other words, all downward arrows represent control actions (commands) while the upward arrows represent feedback. These conventions help to manage complexity and make control relationships and feedback loops easier to recognize.[8] In some cases, simply drawing a control structure diagram can make previously undiscovered flaws painfully obvious. For example, control actions may be provided by entities that don't have the necessary feedback to select safe control actions, feedback may be provided to entities with no ability to do anything about it, multiple controllers may be able to provide conflicting commands to the same component with no ability to detect or resolve the conflict, etc. If you do not catch flaws at this stage though, don't worry—the rest of the STPA method will systematically identify these and other issues.

Common Points of Confusion

A control structure is not a physical model

The hierarchical control structure used in STPA is a functional model, not a physical model like a physical block diagram, a schematic, or a piping and instrumentation diagram. The connections show information that can be sent, such as commands and feedback—they do not necessarily correspond to physical connections. For example, the interactions between the flight crew and air traffic control are not of a physical nature, but they are modeled in a functional control structure.

A control structure is not an executable model

The control structure is not an executable model or a simulation model. In fact, control structures often include components for which executable models do not exist (such as humans). Instead, STPA can be used to carefully derive the necessary behavioral constraints, requirements, and specifications needed enforce the desired system properties. For example, the control structure in *Figure 2.8* does not assume that air traffic control will always sent instructions to the flight crew when needed, that they will always have the capability to send instructions (e.g. that radios will always be operational), that the correct instructions will always be sent, or that the instructions will always be followed by the pilots. It simply indicates that a system was/will be created to allow air traffic control to send instructions to the

[7] A control structure can also contain actuators and sensors, but these are usually added later during the scenario identification step

[8] Control structures can also be drawn left to right if needed, but we are purposely simplifying the discussion here.

flight crew. The next steps in STPA will carefully examine how unsafe behaviors may occur, including instructions that are sent but not received, unsafe instructions that may be sent, etc. Although the control structure itself is not an executable model, the STPA method will produce precise requirements and other outputs that can be used to generate executable models and specifications.

A control structure does not assume obedience

Do not confuse controllers and control actions with obedience. Just because a controller sends a control action does not mean that in practice it will always be followed. Likewise, just because a feedback path is included in a control structure does not mean that in practice the feedback will always be sent when needed or that it will be accurate. The control actions and feedback in a control structure simply indicate that a mechanism will be created to send this information (that is, it will be in the system design). It does not imply or assume anything about how controllers and processes will actually behave in practice. In fact, a major goal of STPA is to analyze the control structure and anticipate how each element might behave in unsafe and potentially unexpected ways.

Use abstraction to manage complexity

One of the biggest challenges in any hazard analysis is managing system complexity. Control structures use abstraction in several ways to help manage complexity. For example, in commercial flights the flight crew might consist of two or three individual pilots. Rather than clutter the control structure right from the beginning with three separate pilots, we can group them together as a flight crew that collectively provides control actions and collects feedback. Similarly, rather than explicitly listing every individual aircraft subsystem, we could begin at a more abstract level by modeling aircraft automation and the physical processes they control as two levels in the control hierarchy.

The principle of abstraction can also be applied to the command and feedback paths in the control structure. Rather than listing each individual button, switch, and lever in the cockpit, we might begin with much broader actions like a climb maneuver. Later, we could refine these broad actions into pitch, thrust, or other commands as appropriate. This principle is especially useful during early development phases when individual commands and sensors are not yet known.

The control action path may contain mechanisms by which the controller acts upon a controlled process (referred to as actuators) and mechanisms by which the controller senses feedback from a controlled process (referred to as sensors). These details are usually abstracted away when initially creating the control structure, but the control structure will be refined to include actuators and sensors later during the scenario creation step.

One additional type of abstraction is used in control structures. Consider the flight commands sent from the flight crew to the aircraft automation. In a remotely piloted UAV application, those commands may need to pass through many different components—from a physical button on the command console, through an embedded system that encodes the command within a digital packet, to a network switch, a radio transmitter, a satellite, and a radio receiver on the UAV. It is not necessary to show all of these detailed steps along the control path in the initial control structure—what matters is that the remote pilot will have some way to send flight commands to the UAV.

In fact, the most efficient way to apply STPA is to begin before those design decisions have been made and before such details are known. The abstract control structure above can be used to begin STPA and identify the requirements and constraints for the communication path and other parts of the system. Then, STPA results can be used to drive the architecture, preliminary and detailed design, make implementation decisions, and refine the control structure. Even if details are known and design decisions have been made, it can be helpful to first apply STPA at a higher abstract level first to provide quicker results and identify broader issues before analyzing more detailed control structure models.

Modeling the control structure

Control structure modeling begins with an abstract control structure and iteratively adds detail. In many cases, the control structure and the control loops within the system may be obvious or can be reused from previous applications. If the control structure is not obvious, the following guidance explains one way to model a functional control structure and add detail. Appendix B contains examples of control structures that have been used in different industries.

One way to begin is to identify basic subsystems needed to enforce the constraints and prevent the hazards identified earlier. For example, we derived hazards and constraints related to insufficient deceleration. These constraints could be enforced by using a wheel braking subsystem, reverse thrust, and other subsystems. *Figure 2.9* shows an initial control structure with these subsystems.

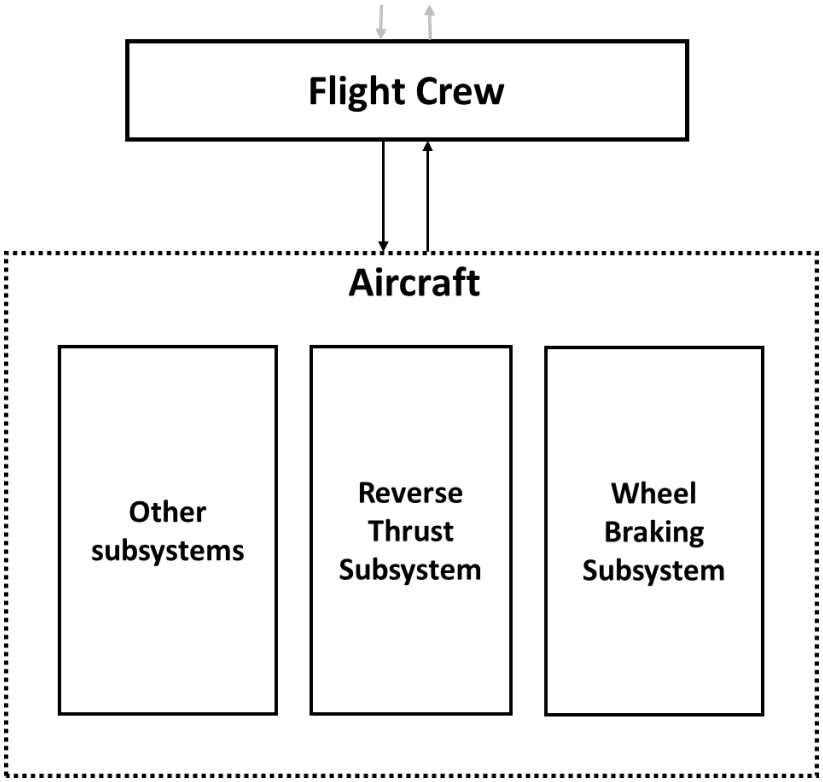


Figure 2.9: Example of a refined control structure with subsystems

The control structure can be refined by defining how each subsystem will be controlled. For the purposes of this example, the wheel braking subsystem will be refined. Will the wheel braking subsystem be controlled directly and manually by the flight crew only? Will control units, automated controllers, or other humans exist that can also control the brakes? If STPA is being applied during later development stages, these answers may be known and obvious. If STPA is being applied during early concept development, then the hazards and the constraints above can be used to guide these decisions. For example, we may include an automated brake controller in the wheel braking subsystem to automatically brake upon landing or rejected takeoff.

Figure 2.10 shows the refined control structure including a Brake System Control Unit (BSCU) within the wheel braking subsystem. Notice that we are carefully "zooming in" to add more detail in the control structure.

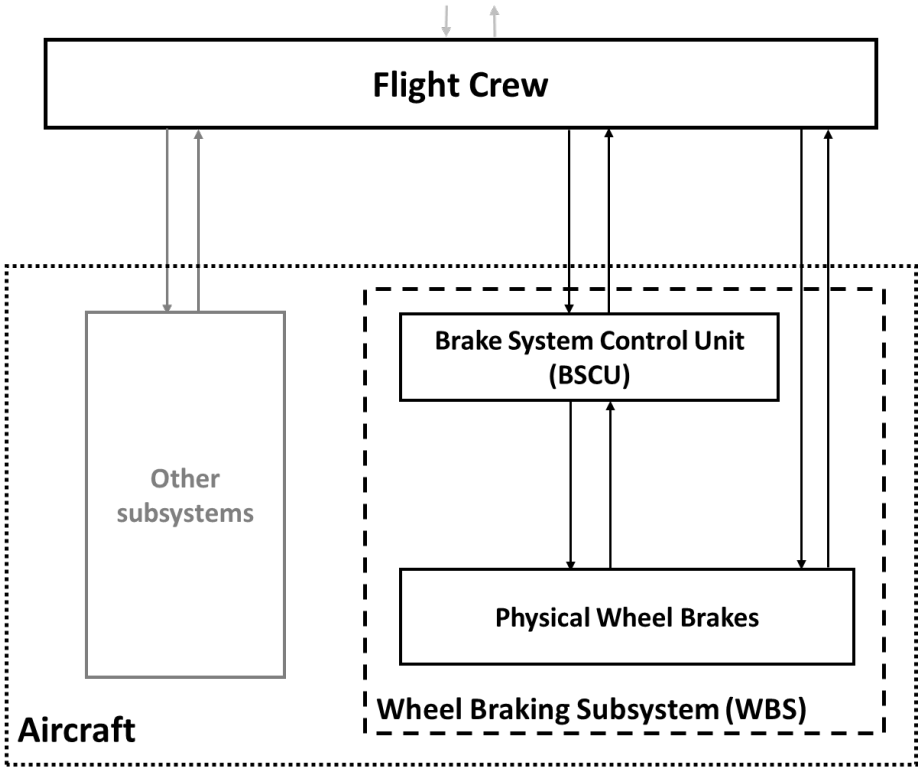


Figure 2.10: Refined control structure with subsystem controllers

During control structure development, responsibilities can be assigned to each control structure entity. These responsibilities are a refinement of the system-level constraints—what does each entity need to do so that *together* the system-level constraints will be enforced? For example, the BSCU might be responsible for automatically pulsing the brakes when braking causes a skid (anti-skid function) while the flight crew may be responsible for deciding when the brakes need to be applied.

Example responsibilities related to wheel braking:

Physical Wheel Brakes

- R-1: Decelerate wheels when commanded by BSCU or Flight Crew [SC-6.1]

BSCU

- R-2: Actuate brakes when requested by flight crew [SC-6.1]
- R-3: Pulse brakes in case of a skid (Anti-skid) [SC-6.2]
- R-4: Automatically engage brakes on landing or rejected takeoff (Autobrake) [SC-6.1]

Flight crew

- R-5: Decide when braking is needed [SC-6.1, SC-6.3]
- R-6: Decide how braking will be done: Autobrake, normal braking, or manual braking [SC-6.1]
- R-7: Configure BSCU and Autobrake to prepare for braking [SC-6.1]
- R-8: Monitor braking and disable BSCU, manually brake in case of malfunction [SC-6.1, SC-6.2]

Control actions for each controller can be defined based on these responsibilities. For example, the flight crew will need the capability to send manual braking control actions to satisfy R-5 and R-6. They will need a way to arm and set the BSCU to satisfy R-6 and R-7. They may need to disarm the BSCU to

satisfy R-8. Figure 2.11 shows a revised control structure with labeled control actions based on the responsibilities.

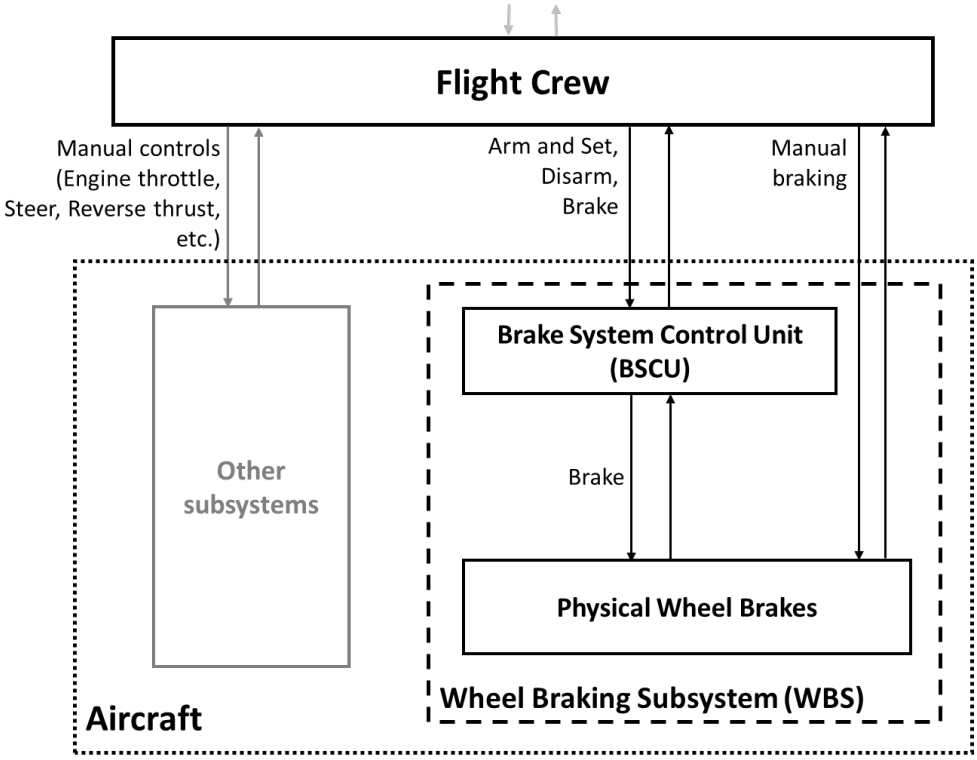


Figure 2.11: Refined control structure after allocation of processes to subsystems

At this point, the controllers and control actions have been identified and labeled but what about feedback? Feedback can be derived from the control actions and responsibilities by identifying the process models that controllers will need to make decisions. Then, feedback and other information needed to form accurate process models can be identified. For example, R-3 specifies that the BSCU will need to pulse the brakes in case of a skid. To do this, the BSCU will need to know that a skid is occurring (information that should be included in the BSCU's process model). What feedback is needed to detect a skid? Wheel speed feedback could be used. Similarly, R-8 specifies that the flight crew may need to disable the BSCU in case of a malfunction. To do this, the flight crew will need to know that the BSCU is malfunctioning (again, information that should be in the process model). What feedback can the flight crew use to detect a malfunction? Feedback about BSCU faults could be provided. To automatically engage the brakes upon landing or RTO (R-4), the BSCU will need to know when the aircraft has landed or when RTO is occurring (should be in the process model). Weight on wheels switches and other inputs could be used to detect landing and RTO conditions. *Table 2.2* below shows how feedback can be derived from the responsibilities.

Table 2.2: Examples showing how feedback can be derived from responsibilities

BSCU Responsibility	Process Model	Feedback
Actuate brakes when requested by flight crew [SC-6.1]	Braking is requested by flight crew	Brake pedal applied
Pulse brakes in case of a skid (Anti-skid) [SC-6.2]	Aircraft is skidding	Wheel speeds Inertial reference unit
Automatically engage brakes on landing or RTO (Autobrake) [SC-6.1]	Aircraft landed Takeoff is rejected	Weight on wheels Throttle lever angle

The control structure can be refined further by using the responsibilities to "zoom in" again and add additional details. For example, the physical wheel brakes are responsible for decelerating the wheels on command (R-1). This could be done with hydraulics. R-6 indicates that the BSCU will need to execute both normal and automatic braking (Autobrake). Two controllers within the BSCU could be used to control these two behaviors: an autobrake controller and a hydraulic controller.

Figure 2.12 shows a refined control structure with feedback labeled and with BSCU internal controllers identified.

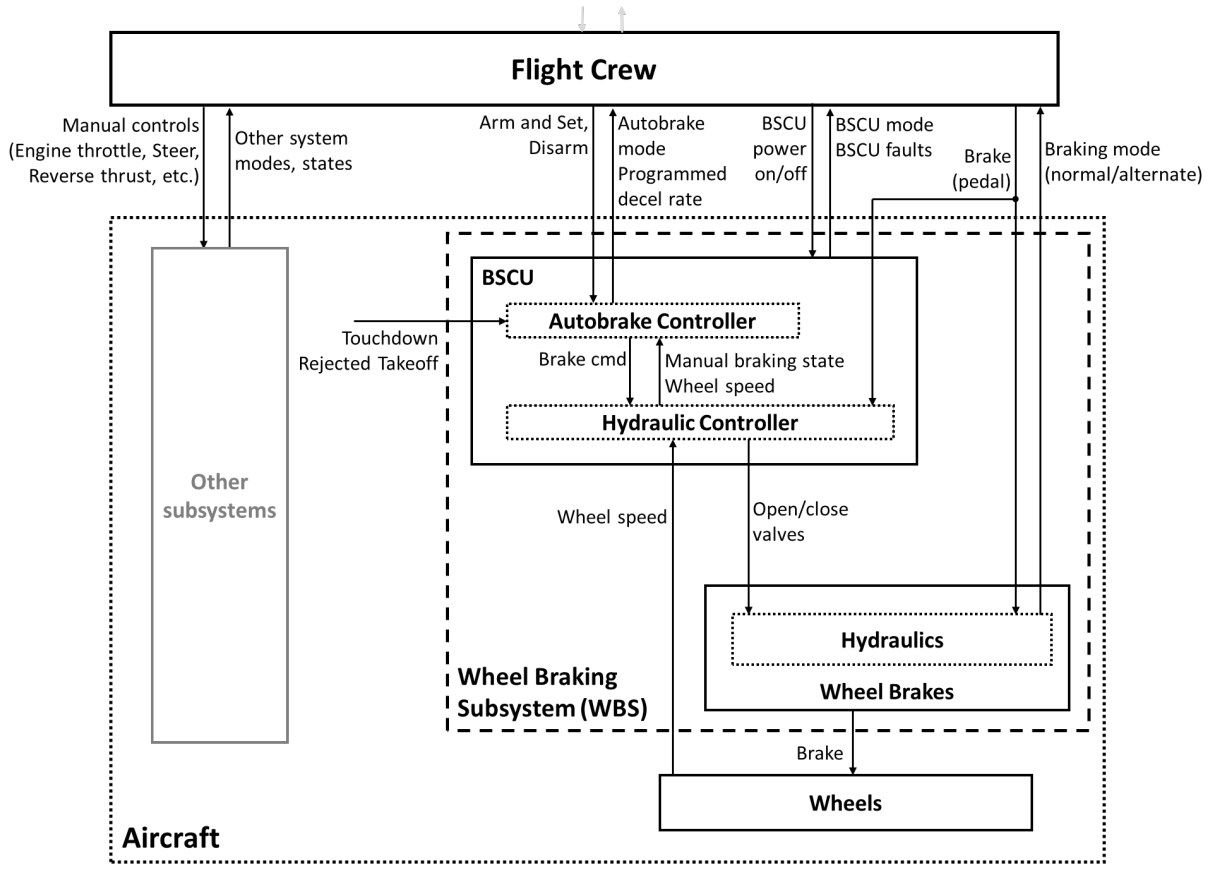


Figure 2.12: Control structure after refinement based on the responsibilities

Common questions when modeling the control structure

Does the control structure need to be complete before proceeding?

If STPA is being applied early before development is finished, some information may not be known and the control structure may be incomplete. The analysis can begin with an incomplete control structure and STPA will help identify potentially missing feedback, controls, and other gaps so the control structure can be refined in parallel with development. The bare minimum needed to begin the next steps is at least one controller, control action, and controlled process. However, in general STPA will be easier and more efficient if relevant information is not intentionally missing from the control structure.

If STPA is being applied during later development stages after decisions about controllers, control actions, and feedback have already been made, then there is no reason to intentionally omit high-level information from the control structure. STPA can be applied to the design as it exists to identify potential flaws like critical feedback that was overlooked.

How specific should the arrow labels be?

Avoid using ambiguous and vague labels in the control structure, such as labeling all control actions as simply "Commands", all feedback as simply "Feedback" or "Status", and controllers as simply "Computer" or "Controller." Labels for control actions should indicate the type of command (if known), like "Open/close valves" and feedback should indicate the type of information being sent (if known), like "Wheel speed". The specific physical medium used to send control actions and feedback is irrelevant at this point—what matters is the *functional* information that can be sent. For example, use "BSCU power on/off" rather than "BSCU button", "Shared bus", or "Encoded digital packets." Similarly, labels for controllers should indicate the functional type of behavior or the role of the controller, not the physical implementation. For example, use controller labels like "Autobrake Controller" rather than "Single Board Computer," and use labels like "Flight Crew" or "Pilot" whether the piloting function is accomplished on board or on the ground.

Should the control structure include all actuators and sensors?

Specific actuators and sensors are not needed to begin STPA and do not need to be included yet (they will be included in a later step). To help manage complexity, it can be helpful to wait until these details need to be considered during the scenario creation step later in STPA. For example, the wheel braking subsystem control actions and feedback in Figure 2.12 may be implemented with different kinds of electromechanical valves or by using some other completely different implementation. What matters at this phase are the types of commands and feedback that might be provided, not the specific implementation (which may or may not be known).

How do physical processes and physical interactions fit into the control structure?

Like any model, a control structure model emphasizes certain aspects of the real world while abstracting or de-emphasizing others. A control structure will emphasize functional relationships and functional interactions, which is very useful for identifying problems like design flaws, requirements flaws, human error, software errors, and even traditional physical component failures. A control structure model does not typically capture purely physical or geometric relationships like physical proximity between components or fire propagation.

The physical processes being controlled are typically specified at the lowest level of the control structure while every level above specifies functional controllers that make decisions and directly or indirectly control the physical processes.

Does the control structure require a linear hierarchy?

No. For some systems, the control structure may appear similar to a ladder with a clear vertical linear hierarchy. Control structures for other systems may include multiple controllers at the same vertical level that do not control each other. Controllers may all control the same physical process, or they may control different processes.

Controllers at the same level may also communicate with each other outside of a control/feedback relationship. Such communication may be represented by horizontal arrows in the control structure diagram. In general, the interactions (arrows) in a control structure can represent any of three categories: Control actions, feedback, and other information.

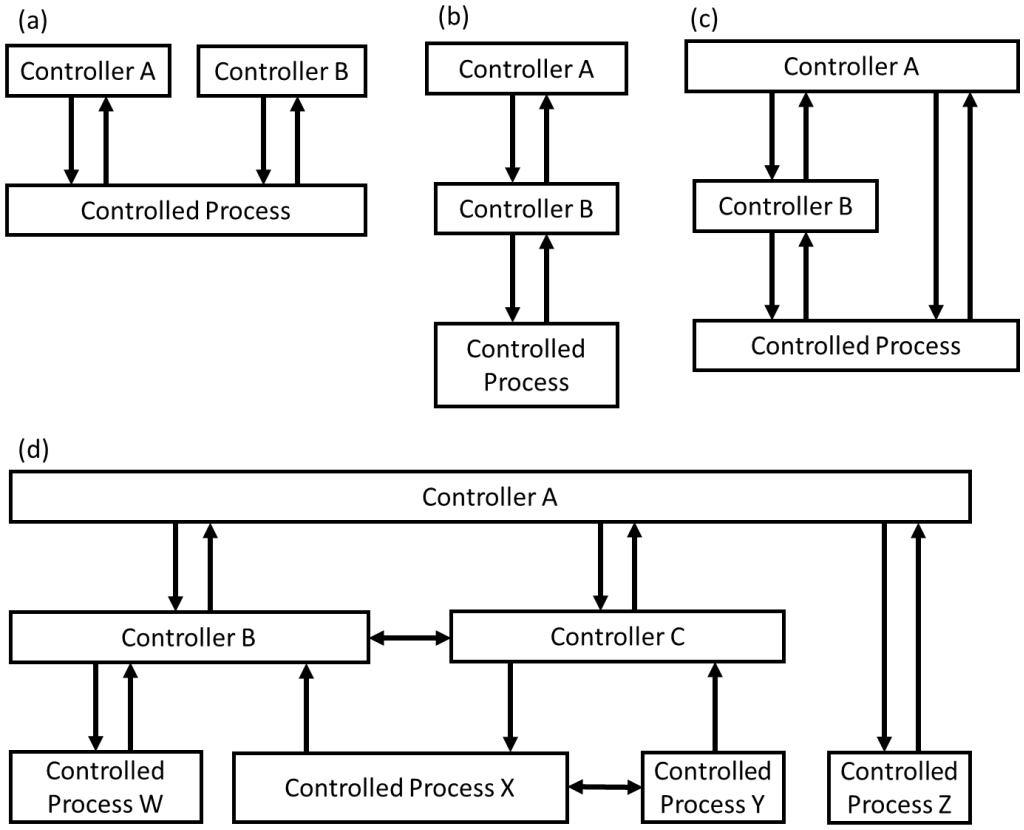


Figure 2.13: Different types of control structures

Figure 2.13 shows a few different types of control structure models, each with different advantages and tradeoffs to consider when designing or analyzing a system. The control structure in (a) shows two controllers operating in parallel to control and monitor a shared process. The controllers do not interact with each other and neither one can control the other (which may lead to loss scenarios that STPA will identify). This type of structure could be advantageous if controllers will need to act quickly or independently of each other—for example, it might be advantageous to provide several operators with the capability to directly abort an operation whenever an unsafe condition is observed without going through a chain of command. However, this structure can present coordination challenges like diffusion of responsibility and assumptions about the actions of other controllers (later steps will analyze these and other problems in more detail).

The control structure in (b) shows a linear control hierarchy such as a clear chain of command. This structure may help address the coordination issue and clarify the difference in responsibilities for each

controller, but may present challenges in terms of reaction time for Controller A's feedback and control actions and is more dependent on the operation of Controller B.

Control actions and feedback may "skip" over levels—there is no model limitation that prevents controllers from interacting beyond one level above and below. For example, high-level controllers may be able bypass other controllers and directly control or monitor lower-level process as in (c). Controller A could represent pilots that normally use automation in Controller B to actuate the brakes. However, in an emergency the pilots may be able to bypass Controller B and directly (manually) actuate the brakes.

More complex structures like (d) are also possible. There is no requirement for a 1-to-1 mapping between controllers and processes. A controller may control one or more processes, and a process may be controlled by zero or more controllers. Typically, each control action path will be paired with a parallel feedback path but not always. For example, Controller C may not be able to control Process Y directly. Instead, it may control Process X and monitor the effect of X on Process Y.

How do I figure out who controls who?

In many cases the control relationship is simple and obvious, such as a supervisor giving instructions to a subordinate or a computer sending commands to open/close a valve. Control relationships can also be less direct, such as managers assigning priorities, airlines providing standard operating procedures, etc. These are all different forms of control.

As mentioned earlier, control is different from obedience. Just because a control relationship exists does not mean that the control actions will always be executed and performed adequately. In fact, we may not even want all control actions to be obeyed all the time; control actions may potentially be disregarded for good reasons when required to prevent a loss in some unexpected situation. STPA does not assume obedience and these problems will be carefully examined in later steps.

Similarly, the ability to trigger a response or influence behavior of another component does not automatically imply control. A control action to open a valve may trigger the valve to open, but conversely a feedback signal indicating high temperature may trigger a controller to turn on a fan. The ability to influence or respond is not sufficient to distinguish control actions from feedback.

Control involves making purposeful decisions to achieve goals. Feedback can be provided without being aware of (or responsible for) a particular higher-level goal, but control actions are always provided to achieve a goal. Control also typically involves supervision—monitoring lower-level entities, having better capability or information to evaluate their behaviors or impacts, and guiding them or intervening to ensure higher-level goals are achieved.

The control hierarchy is closely related to the hierarchy of goals and responsibilities. The automated BSCU controller above may have very narrow responsibilities related to triggering brakes when a landing is detected. The flight crew has higher-level responsibilities like deciding when to land and higher-level goals like avoiding a rough landing. The BSCU may be a necessary but insufficient element in achieving these higher-level flight crew goals—the flight crew must supervise the BSCU and many other elements to achieve the higher-level goals. Similarly, air traffic control has even higher-level responsibilities like ensuring minimum separation between multiple aircraft and higher-level goals like maximizing throughput across many flights. A single flight crew is only one element used by air traffic control to achieve the goals, and the flight crew might not even be aware of how they are contributing to those higher-level goals or if those higher-level goals are being achieved by air traffic control.

Finally, control is closely related to authority. Consider the relationship between air traffic control and pilots. Air traffic control has authority over pilots and provides instructions and clearances that pilots generally must obey. Pilots do not have authority over air traffic control, and they cannot simply order air traffic control to obey. Pilots *can* declare an emergency and air traffic control will need to respond appropriately to that feedback—just because pilots can trigger a response does not imply

control. Pilots can also disobey air traffic control instructions when necessary to ensure safety of the flight, but obedience is also different from control. Pilots do have final and direct authority over their aircraft, but they do not have final authority over ATC and they do not manage the overall airspace. ATC generally controls aircraft indirectly by issuing instructions and clearances, not the other way around. Chapter 5 contains further discussion about control structures and control relationships as they relate to organizational and social analysis.

Although beginners sometimes struggle with these distinctions, with more experience it quickly becomes easy and natural. It can also be helpful to realize that mistakes about who controls whom in the control structure usually do not have a significant impact on the results of the analysis. For example, suppose control action X is mischaracterized as feedback X. Because it is characterized as feedback, the step that identifies unsafe control actions will not consider how a missing or delayed control action X might lead to a hazard. However, the next step examines potential feedback problems and will identify the same scenarios when considering how missing or delayed feedback X might lead to a hazard.

Do I need to document anything other than the control structure diagram?

It is good practice to document any additional information that is helpful to understand the control structure. Additional information about controllers such as a basic description, purpose, special functionality, controller responsibilities, process models, etc. should be documented for a completely specified control structure. Clarifying information about control actions and feedback can also be helpful, particularly when important aspects may not be clear from short labels. The same is true for controlled processes. In general, any important clarifying information or assumptions should be clearly documented along with the control structure diagram.

What should I look for when reviewing a control structure?

The following tips can help find common mistakes in a control structure:

Tips to prevent common mistakes in a control structure

- Ensure labels describe functional information that is sent, not a specific physical implementation.
- Avoid ambiguous and vague labels like simply "Command" or "Feedback" when the type of information is known.
- Check that every controlled physical process is controlled by one or more controllers (not always required, but often indicates a mistake).
- Review responsibilities (including traceability) for conflicts and gaps.
- Check that control actions needed to satisfy the responsibilities are included.
- Check that feedback needed to satisfy the responsibilities is included. (optional if applied early in concept development when feedback is unknown; later steps can identify missing feedback)

Identifying Unsafe Control Actions

Once the control structure has been modeled, the next step (shown in Figure 2.14) is to identify Unsafe Control Actions.

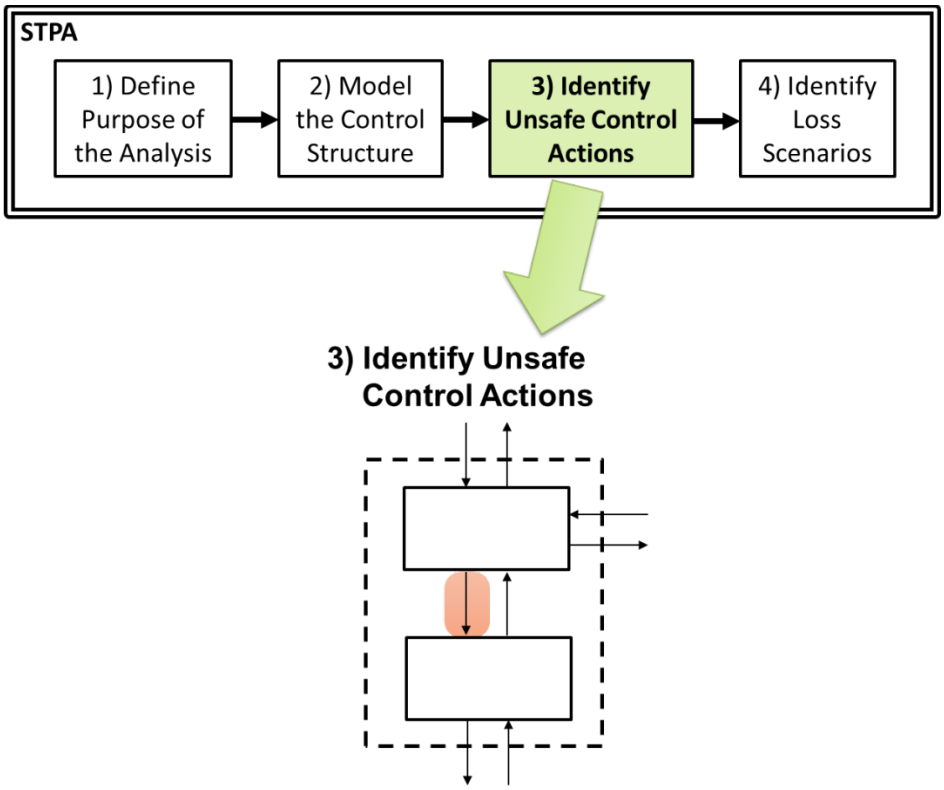


Figure 2.14: Identify Unsafe Control Actions

Definition: An Unsafe Control Action (UCA) is a control action that, in a particular context and worst-case environment, will lead to a hazard.[9]

Table 2.3 below shows examples of UCAs for the BSCU controller. A larger set of UCAs for the wheel braking subsystem can be found in Appendix C.

[9] The term "unsafe" refers to the hazards identified in STPA. As discussed earlier, hazards can include issues related to loss of human life or injury (traditional safety) but they can also be defined much more broadly to include other losses like a mission loss, loss of performance, environmental losses, etc.

Table 2.3: Examples of Unsafe Control Actions for the BSCU (partial example)

Control Action	Not providing causes hazard	Providing causes hazard	Too early, too late, out of order	Stopped too soon, applied too long
Brake	UCA-1: BSCU Autobrake does not provide the Brake control action during landing roll when the BSCU is armed [H-4.1]	UCA-2: BSCU Autobrake provides Brake control action during a normal takeoff [H-4.3, H-4.6] UCA-5: BSCU Autobrake provides Brake control action with an insufficient level of braking during landing roll [H-4.1] UCA-6: BSCU Autobrake provides Brake control action with directional or asymmetrical braking during landing roll [H-4.1, H-4.2]	UCA-3: BSCU Autobrake provides the Brake control action too late (>TBD seconds) after touchdown [H-4.1]	UCA-4: BSCU Autobrake stops providing the Brake control action too early (before TBD taxi speed attained) when aircraft lands [H-4.1]

There are four ways a control action can be unsafe (represented in the columns above):

1. Not providing the control action leads to a hazard.
2. Providing the control action leads to a hazard.
3. Providing a potentially safe control action but too early, too late, or in the wrong order
4. The control action lasts too long or is stopped too soon (for continuous control actions, not discrete ones).

Consider UCA-2:

> UCA-2: BSCU Autobrake provides Brake command during a normal takeoff [H-4.3, H-4.5]

This UCA is unsafe because it can lead to H-4.3: Deceleration occurs after the V1 point during takeoff and H-4.5: Acceleration is insufficient during takeoff. Every UCA can be traced to one or more hazards (or sub-hazards), and it is good practice to document the traceability in brackets at the end of every UCA.

UCAs should specify the context in which the control action is unsafe. The context is critical. Suppose you know that the BSCU on an aircraft can provide a Brake command. Could that command be unsafe? It's impossible to assess without considering the context. UCA-2 contains the context "during a normal takeoff", which is what makes the control action unsafe.

If a control action were always unsafe, then it probably would not have been designed into the system to begin with. Every UCA must specify under what conditions (in what context) the control action is unsafe. Then we can eliminate those instances from the system design or find ways to mitigate them. Any relevant context can be referenced in a UCA, including environmental conditions, controlled process states, states of the controller, previous actions by the controller (e.g. repetitive actions), states of other

controllers, previous actions by others, simultaneous or conflicting actions, parameters or properties of the control action (e.g. a particular braking rate being programmed), or any other relevant conditions. Using words like “when”, “while”, or “during” in UCA construction is often helpful in developing the context.

A UCA contains five parts:

UCA-2: BSCU Autobrake provides Brake command during a normal takeoff [H-4.3]
<Source> <Type> <Control Action> <Context> <Link to Hazards>

The first part is the controller that can provide the control action. The second part is the type of unsafe control action (provided, not provided, too early or too late, stopped too soon or applied too long). The third part is the control action or command itself (from the control structure). The fourth part is the context discussed above, and the last part is the link to hazards (or sub-hazards).

UCAs are often written with each part in the same order shown above, but in some cases it may be clearer or more natural to use a different ordering. The ordering is not critical. The key point is that UCAs contain these five parts.

Common questions about UCAs

Does a UCA guarantee that a hazard will always result?

No. In a best-case scenario, UCA-2 might occur early during takeoff, the pilots recognize that UCA-2 has happened, they immediately disable the BSCU, and there is no accident. In a worst-case scenario, the pilots might not be able to react and recover in time, their actions to disable the BSCU might be ineffective, there is a significant tail-wind and UCA-2 occurs right after the V1 decision speed, or other factors arise and the aircraft runs off the runway (H-4). The goal of STPA is not to argue whether the best-case or worst-case scenarios are more likely or to make assumptions about pilot capabilities and response. When engineering the BSCU we’d still like to prevent it from sending brake commands during normal takeoffs regardless of whether the flight happens to occur in a best-case or a worst-case environment. The goal of this step is simply to identify the behaviors that should be prevented. STPA is a worst-case analysis method not a best-case, average-case, or most-likely-case method.

Can I identify UCAs when we already have safeguards in place?

Systems may include safeguards like protective features, redundancies, and backup systems specifically meant to prevent unsafe control actions from causing hazards. For example, some might argue that UCA-1: “BSCU Autobrake does not provide the Brake control action during landing roll when the BSCU is armed” cannot lead to a hazard because an independent alternate braking system is included in the design that effectively bypasses the BSCU and allows manual braking at any time. In a best-case scenario, the safeguards will operate as intended, will be effective and sufficient, and the hazard may be avoided. However, in a worst-case scenario the safeguards may not operate as intended, may not be sufficient, or they may not be effective for the situation at hand. Similar to the reasoning above, STPA is a worst-case analysis method and we cannot omit UCAs when a safeguard exists.

Another way to understand this reasoning is to realize that we’d like to prevent UCAs *even if safeguards do exist*. For example, even though an alternate braking system may be included in the design we wouldn’t intentionally design the BSCU to provide UCAs. We’d like to make sure the BSCU Autobrake will provide the proper brake control actions (e.g. to prevent UCA-1, etc.) even when safeguards like an alternate braking system are available.

In fact, STPA is ideally applied early before safeguards are known and incorporated into the design. STPA identifies UCAs that must be prevented, and the UCAs are then used to derive functional requirements and make design decisions to prevent or mitigate the UCAs. After potential unsafe

behavior is identified, then the specific design features can be created and safeguards added (if the design does not already exist) or the adequacy of existing design decisions and safeguards can be determined (if the design already exists).

The last two types of unsafe control actions are both about timing. What's the difference?

The third type of UCA is about control actions provided at the wrong time—too early, too late, or out of order. The fourth type of unsafe control action only applies to control actions with a duration, that is, continuous or non-discrete control actions.

First, suppose the Brake command provided by BSCU Autobrake is implemented as a continuous control action. In other words, the brakes are applied when Autobrake starts providing the Brake control action, and the brakes continue to be applied until Autobrake stops providing the Brake control action. The initial timing of the Brake control action can cause a hazard:

> UCA-3: BSCU Autobrake provides the Brake control action too late (>TBD seconds) after touchdown [H-4.1]

The Brake control action could also be provided correctly during a landing and on-time without delay (within TBD seconds), but Autobrake then immediately stops providing the control action and the brakes immediately cease to be applied. The braking was stopped too soon to be effective. This behavior could cause H-4.1 even though the control action was originally provided in the right situation and on time, as reflected in UCA-4:

> UCA-4: BSCU Autobrake stops providing the Brake control action too early (before TBD taxi speed attained) when aircraft lands [H-4.1]

Now let's consider an alternative situation: the BSCU Autobrake provides two separate discrete control actions to Start Braking and to Stop Braking. The individual control actions do not have a meaningful duration, so "stopped too soon / applied too long" is not applicable. Instead, we'd consider how the either control action could be provided too early or too late:

> UCA-3: BSCU Autobrake provides the Start Brake control action too late (>TBD seconds) after touchdown [H-4.1]
>
> UCA-4: BSCU Autobrake provides the Stop Brake control action too early (before TBD taxi speed attained) when aircraft lands [H-4.1]

Notice that the same issues are covered by the analysis whether control actions are continuous or discrete.

Do I need to identify exactly one UCA for each type of unsafe control action?

No. Although all four UCA types should be considered, they may not all be applicable in every case. It is also possible to identify several UCAs of a single type, as demonstrated by UCA-2, UCA-5, and UCA-6 above. In general, each category may contain 0, 1, 2, or more UCAs.

Are there more than four types of unsafe control? Is another category needed?

These four categories are provably complete—there is no other category of unsafe control action. However, there are subcategories. For example, there are several ways the second type of UCA could occur:

1. Not providing the control action leads to a hazard
2. Providing the control action leads to a hazard
    a. Consider contexts in which the control action may never be safe

b. Consider contexts in which the control action has an incorrect parameter (e.g. setting an incorrect emergency frequency on a radio)
c. Consider contexts in which an insufficient or excessive control action may be unsafe (e.g. providing insufficient or excessive braking commands)
d. Consider contexts in which the direction of the control action may be unsafe (e.g. providing turn left instead of turn right commands)
e. Consider contexts in which the control action has already been provided (e.g. repetitive, oscillatory, intermittent control actions)
f. Consider contexts in which the control action is provided too quickly or too slowly (e.g. ramp rate, frequency, etc.)
g. Etc.

3. Providing a potentially safe control action but too early, too late, or in the wrong order
4. The control action lasts too long or is stopped too soon

Case (b) is only relevant for control actions that include one or more parameters. For example, a BSCU Brake control action is not typically a simple binary on/off control action. The BSCU controls braking by commanding the specific amount of braking to be applied. Recall UCA-5 above, which states that even though the BSCU may correctly provide a brake command during landing roll, it could still be unsafe if the BSCU commands an *insufficient* level of braking. UCA-6 exists because the braking commands could be sent asymmetrically causing the aircraft to turn in a wrong direction. Whenever control actions can specify one or more parameters, it is important to consider how the parameters may be insufficient, excessive, in a wrong direction, or otherwise unsafe.

I just identified an important UCA that isn't related to any system-level hazards. What do I do?

Every UCA must be traceable to one or more system-level hazards. If you identify a UCA that does not relate to one of the identified hazards, you could be missing a hazard. Identify the system-level state or condition caused by the UCA and consider adding a new hazard or revising the existing set of hazards to include the new state or condition. STPA is iterative and need not be performed in a strictly linear fashion—earlier results can be updated as the analysis progresses and as more information becomes available.

Where is the outcome or result of the UCA described?

Every UCA should reference the hazards or sub-hazards it can lead to, which captures system-level effects and outcomes of the UCA. For many systems, this link between a UCA and hazards is clear and obvious. For example, the BSCU providing insufficient braking during landing roll is clearly related to H-4.1: Insufficient deceleration upon landing.

However, in some cases the link between a UCA and the hazards may be more complex or unintuitive. It is good practice to document any special reasoning behind UCAs and their relationship to the hazards, particularly in more complex applications where it is not obvious. The reasoning can sometimes be documented by adding a few words to the UCA, but this can be difficult if the reasoning is not simple. A general solution is to document comments for each UCA as needed. UCA comments may describe the rationale for the UCA, how the UCA can lead to hazards, any assumptions the UCA is based on, the effects or results of the UCA, or other information needed to understand the UCA and how it can lead to hazards.

Be careful not to confuse the UCA context with UCA outcomes, results, or other reasoning. One common mistake is to omit the UCA context and instead include the UCA result. For example:

Correct UCA: BSCU Autobrake provides Brake command during normal takeoff [H-4.3]

Incorrect UCA: BSCU Autobrake provides Brake command resulting in a collision

If the UCA context (the current state or condition) is not identified, the next steps that create requirements and identify scenarios will be difficult or impossible to perform. Every UCA must contain the UCA context. For clarity, it may also contain the result.

Should I specify process model flaws in the UCA context?

Be careful not to confuse a UCA with a cause of a UCA. The UCA context should specify the actual (true) state or condition that would make the control action unsafe, not a particular controller process model or belief (which may or may not be true). The next step in STPA will identify causes of UCAs, such as process model flaws and other factors. For example:

Correct UCA: BSCU Autobrake provides Brake command during normal takeoff

Incorrect UCA: BSCU Autobrake provides Brake command when it incorrectly believes the aircraft is landing

Identifying Human UCAs

The same approach can be applied to human controllers to identify UCAs. For example, consider the flight crew control action to power off the BSCU. *Table 2.4* shows some examples of the corresponding flight crew UCAs for this command. Additional example UCAs for the flight crew related to wheel braking can be found in Appendix C.

Table 2.4: Example Unsafe Control Actions for the Flight Crew (partial example)

Control Action	**Not providing causes hazard**	**Providing causes hazard**	**Too early, too late, out of order**	**Stopped too soon, applied too long**
Power Off BSCU	UCA-1: Crew does not provide BSCU Power Off when abnormal WBS behavior occurs [H-4.1, H-4.4, H-7]	UCA-2: Crew provides BSCU Power Off when Anti-Skid functionality is needed and WBS is functioning normally [H-4.1, H-7]	Crew powers off BSCU too early before Autobrake or Anti-Skid behavior is completed when it is needed [H-4.1, H-7]	N/A

The last column is considered N/A (not applicable) because the Power Off control action does not have a duration in this case. The crew simply provides Power On and Power Off commands to toggle the state of the BSCU. Issues related to the BSCU remaining powered off for too long will be captured when considering the Power On command not provided or provided too late. Note that humans are treated in the same way as other types of system components and can be easily integrated into the overall analysis.

Tips to prevent common mistakes

The following tips will help prevent common mistakes when identifying UCAs:

Tips to prevent common mistakes when identifying UCAs

- Ensure every UCA specifies the context that makes the control action unsafe.
- Ensure UCA contexts specify the actual states or conditions that would make the control action unsafe, not potential beliefs about the actual states.
- Ensure the UCA contexts are defined clearly.
- Ensure the UCA contexts are included and not replaced by future effects or outcomes.
- Ensure traceability is documented to link every UCA with one or more hazards.
- Review any control action types assumed to be N/A, and verify they are not applicable.
- For any continuous control actions with a parameter, ensure that excessive, insufficient, and wrong direction of the parameters are considered.
- Ensure any assumptions or special reasoning behind the UCAs are documented

Defining Controller Constraints

Definition: A controller constraint specifies the controller behaviors that need to be satisfied to prevent UCAs

Once UCAs have been identified, they can be translated into constraints on the behavior of each controller. For example, when analyzing BSCU control actions we determined that the BSCU providing a Brake control action during a normal takeoff could lead to a hazard. Therefore, the BSCU must not provide the Brake control action in that context. In general, each UCA can be inverted to define constraints for each controller. Table 2.5 shows the BSCU behavioral constraints that can be derived from the UCAs in *Table 2.4*.

Table 2.5: Example Controller Constraints for BSCU (incomplete)

Unsafe Control Actions	**Controller Constraints**
UCA-1: BSCU Autobrake does not provide the Brake control action during landing roll when the BSCU is armed [H-4.1]	C-1: BSCU Autobrake must provide the Brake control action during landing roll when the BSCU is armed [UCA-1]
UCA-2: BSCU Autobrake provides Brake control action during a normal takeoff [H-4.3, H-4.5]	C-2: BSCU Autobrake must not provide Brake control action during a normal takeoff [UCA-2]
UCA-3: BSCU Autobrake provides the Brake control action too late (>TBD seconds) after touchdown [H-4.1]	C-3: BSCU Autobrake must provide the Brake control action within TBD seconds after touchdown [UCA-3]
UCA-4: BSCU Autobrake stops providing the Brake control action too early (before TBD taxi speed attained) during landing roll [H-4.1]	C-4: BSCU Autobrake must not stop providing the Brake control action before TBD taxi speed is attained during landing roll [UCA-4]
UCA-5: BSCU Autobrake provides Brake control action with an insufficient level of braking during landing roll [H-4.1]	C-5: BSCU Autobrake must not provide less than TBD level of braking during landing roll [UCA-5]
UCA-6: BSCU Autobrake provides Brake control action with directional or asymmetrical braking during landing roll [H-4.1, H-4.2]	C-6: BSCU Autobrake must not provide directional or asymmetrical braking during landing roll [UCA-6]

Inputs and Outputs for Control Action Analysis

Figure 2.15 summarizes the inputs and outputs for the control action analysis.

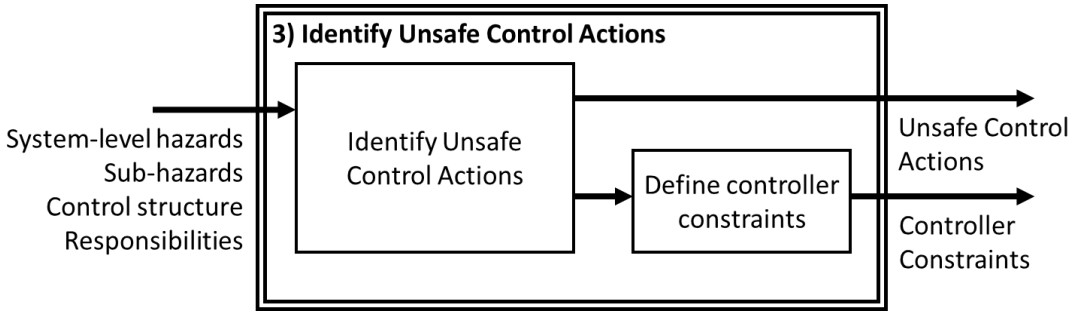


Figure 2.15: Overview of control action analysis

Identifying loss scenarios

Once unsafe control actions have been identified, the next step (shown in Figure 2.16) is to identify loss scenarios.

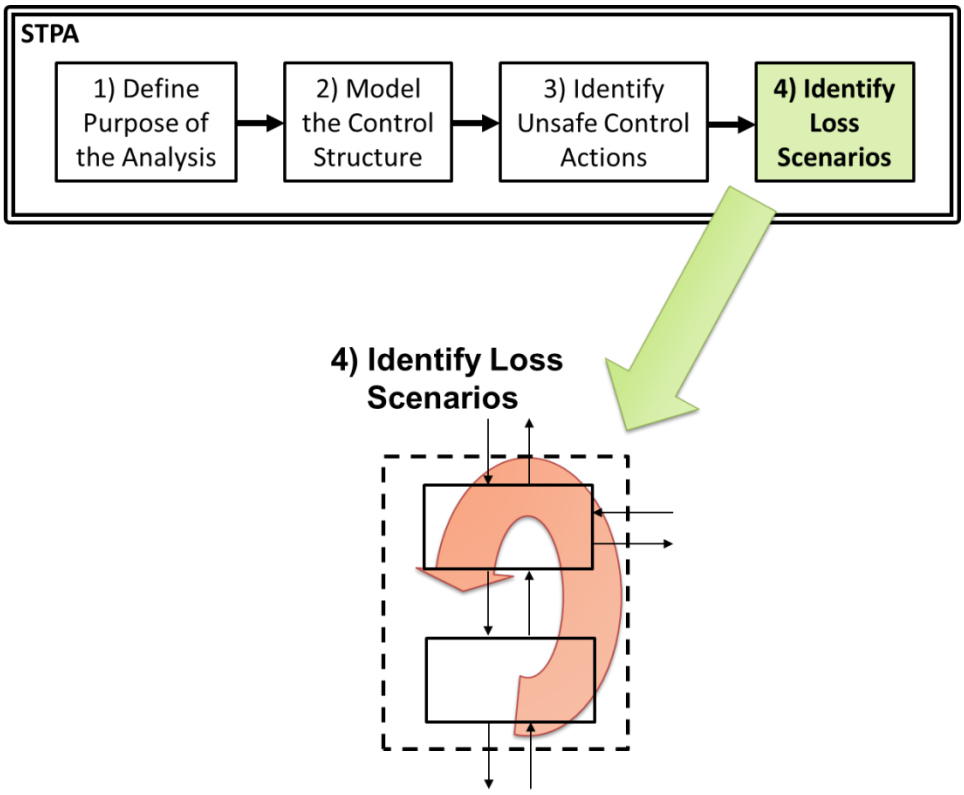


Figure 2.16: Identify loss scenarios

Definition: A loss scenario describes the causal factors that can lead to the unsafe control actions and to hazards.

Two types of loss scenarios must be considered, as shown in Figure 2.17:

a) Why would Unsafe Control Actions occur?
b) Why would control actions be improperly executed or not executed, leading to hazards?

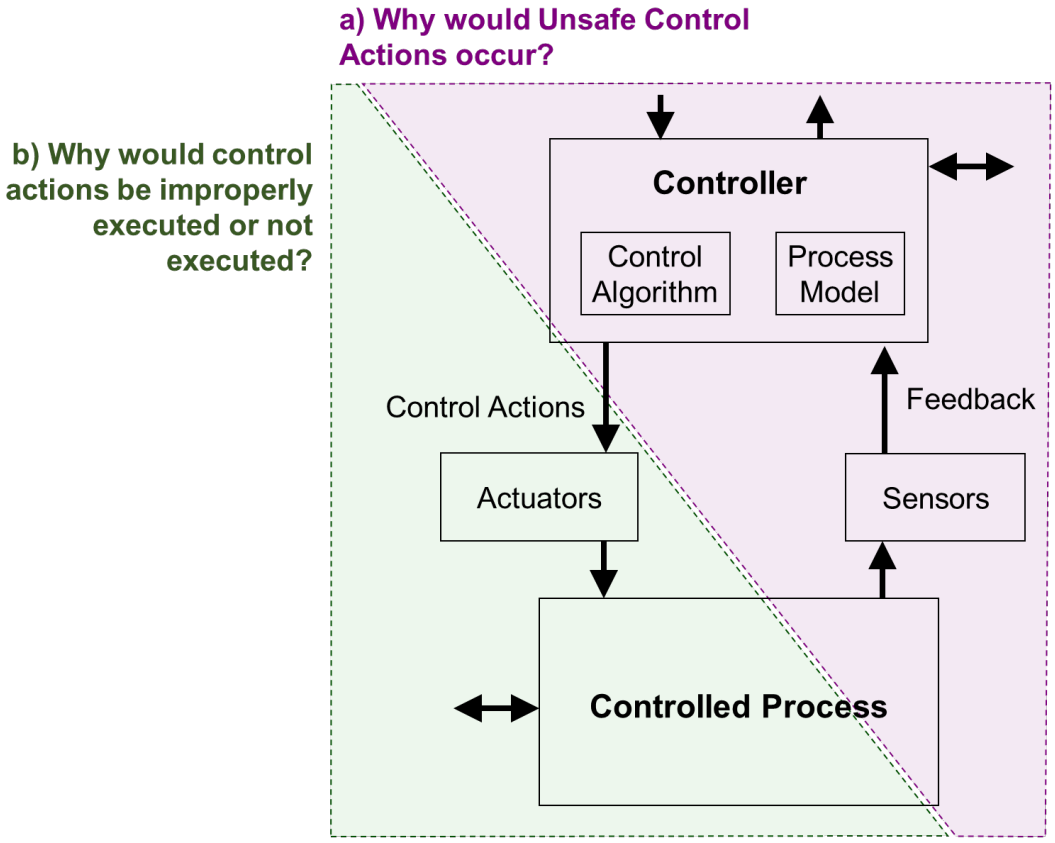


Figure 2.17: Two types of scenarios that must be considered

Notice that Figure 2.17 includes sensors and actuators. Up to this point in the analysis, we have considered the control actions and feedback that may exist but we have not yet examined how the feedback is measured or detected (e.g. with sensors) or how the control actions are executed (e.g. with actuators). Because scenarios identify the specific causes of unsafe control and feedback, it is helpful to refine the control structure to include sensors and actuators.

a) Identifying scenarios that lead to Unsafe Control Actions

This type of scenario can be created by starting with a UCA and working backward to explain what could cause the controller to provide (or not provide) that control action. In general, scenarios that lead to UCAs may include (but is not limited to):

- Failures related to the controller (for physical controllers)
  - Physical failure of the controller itself
  - Power failure
  - Etc.
- Inadequate control algorithm
  - Flawed implementation of the specified control algorithm
  - The specified control algorithm is flawed

    - The specified control algorithm becomes inadequate over time due to changes or degradation
- Unsafe control input
    - UCA received from another controller (already addressed when considering UCAs from other controllers)
- Inadequate process model
    - Controller receives incorrect feedback/information
    - Controller receives correct feedback/information but interprets it incorrectly or ignores it
    - Controller does not receive feedback/information when needed (delayed or never received)
    - Necessary controller feedback/information does not exist

To create scenarios that involve unsafe control actions, we must consider factors shown in Figure 2.18 starting with the unsafe controller behavior that caused the UCA.

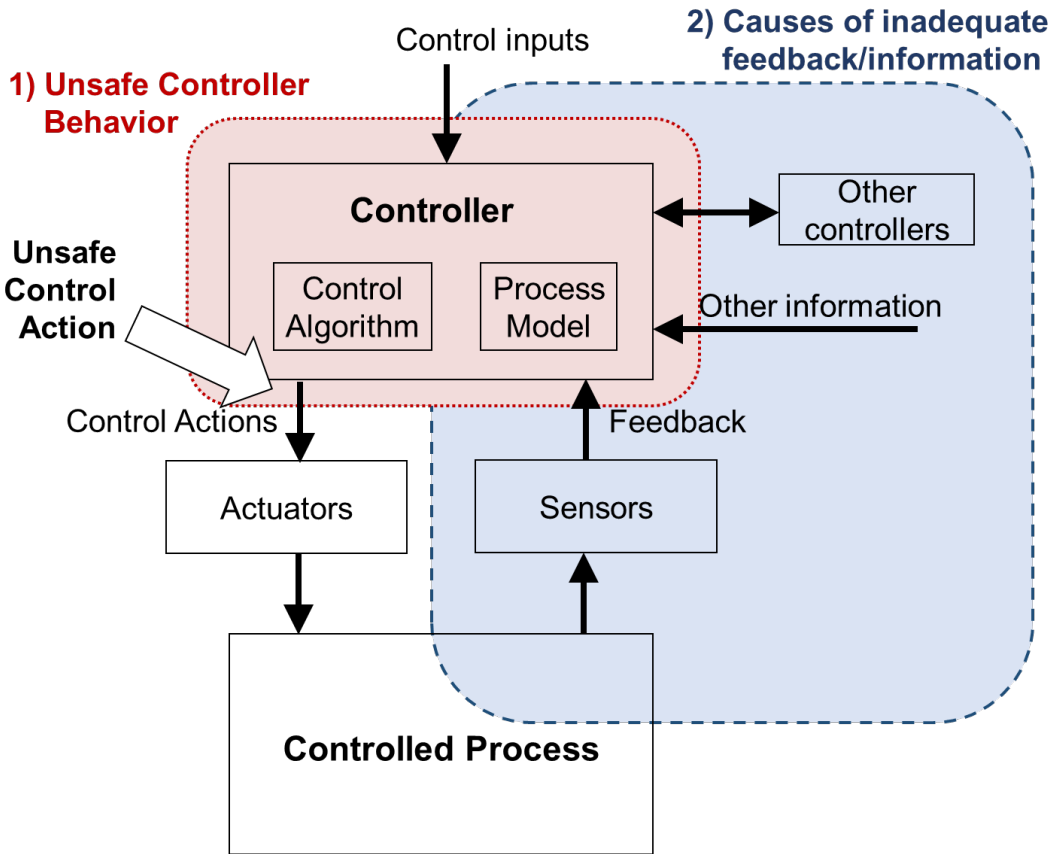


Figure 2.18: Unsafe Control Actions can be caused by (1) unsafe controller behavior and (2) inadequate feedback and other inputs

1) Unsafe controller behavior

There are four general reasons why a controller might provide (or not provide) a control action that is unsafe:

- Failures involving the controller (for physical controllers)
- Inadequate control algorithm
- Insafe control input (from another controller)
- Inadequate process model

For physical controllers, a UCA may occur due to a failure related the controller. For example, the BSCU may not provide the brake command because the BSCU controller fails, because the BSCU power fails, etc. To identify these scenarios, start with a UCA, identify the controller, and identify physical failures related to the controller that can explain the UCA. For example:

> UCA-1: BSCU Autobrake does not provide the Brake control action during landing roll when the BSCU is armed [H-4.1]
>
> Scenario 1 for UCA-1: The BSCU Autobrake physical controller fails during landing roll when BSCU is armed, causing the Brake control action to not be provided [UCA-1]. As a result, insufficient deceleration may be provided upon landing [H-4.1]

An inadequate control algorithm can also cause a UCA. A control algorithm specifies how control actions are selected based on the controller's process model, previous control inputs and outputs, and other factors. For human controllers, the control algorithm is sometimes called decision-making and it may be shaped by different factors like training, procedures, and past experience.

To identify these scenarios, start with a UCA and identify how the control algorithm may cause the UCA. For example:

> BSCU Autobrake example:
>
> UCA-3: BSCU Autobrake provides the Brake control action too late (>TBD seconds) after touchdown [H-4.1]
>
> Scenario 1 for UCA-3: The aircraft lands, but processing delays within the BSCU result in the Brake control action being provided too late [UCA-3]. As a result, insufficient deceleration may be provided upon landing [H-4.1]
>
> Human Crew example:
>
> Crew-UCA-1: Crew does not provide BSCU Power Off when abnormal WBS behavior occurs [H-4.1, H-4.4]
>
> Scenario 1 for Crew-UCA-1: Abnormal WBS behavior occurs and a BSCU fault indication is provided to the crew. The crew does not power off the BSCU [Crew-UCA-1] because the operating procedures did not specify that the crew must power off the BSCU upon receiving a BSCU fault indication.

In general, flaws in the control algorithm can stem from:

- Flawed implementation of the specified control algorithm
- The specified control algorithm is flawed
- The specified control algorithm becomes inadequate over time due to changes or degradation

The specific reasons for each of these types of flaws may be dependent on the application being studied.

One common control algorithm flaw occurs when the control algorithm assumes that previous control actions have been executed properly. This flaw is especially relevant when there is no feedback to indicate whether the control action was successful. For example, the BSCU Autobrake may not provide the Brake control action because it incorrectly assumes that a previous Brake control action was successful and that the aircraft was already braking.

To include scenarios related to security, one additional possibility needs to be considered here: identify if and how the control algorithm flaw could be introduced by an adversary.

Unsafe control inputs from other controllers can also cause UCAs. These can be found during the previous step when identifying Unsafe Control Actions for other controllers.

Finally, inadequate process models can cause Unsafe Control Actions. As explained above, process models represent the controller's internal beliefs that are used by the control algorithm to determine control actions. Process model flaws occur when a controller's process model does not match reality. Process model flaws may occur because:

- Controller receives incorrect feedback/information
- Controller receives correct feedback/information but interprets it incorrectly or ignores it
- Controller does not receive feedback/information when needed (delayed or never received)
- Necessary controller feedback/information does not exist

These problems could arise in many different ways depending on the application. Incorrect feedback/information might be received by the controller, including conflicting information that cannot be resolved or conflicts that may be resolved incorrectly. Correct feedback/information can be received but ignored because the controller is disabled, turned off, busy with another task, is missing a process model with the necessary conditions to update, or some other reason. Controller interpretation problems may occur if the process model is updated incorrectly, the wrong process model is updated, the feedback/information was thought to represent something else, or other errors. Feedback/information may not be received when needed if the feedback is never received—especially if the controller may assume a default value in lieu of feedback—or if feedback is delayed, including information received out of order. Finally, the necessary feedback/information may not exist in the control structure or in the design, resulting in an inadequate process model.

To identify these scenarios, start with a UCA and identify the controller process models that could cause the UCA. Consider beliefs about current states or modes, previous states, capabilities, dynamic behavior[10], previous behaviors or actions, future states or behaviors (predictions), beliefs about the process currently being controlled, other controlled processes, other controllers in the system (especially beliefs needed for coordination), actuators, sensors, or other relevant aspects of the system or environment.

Once the relevant process models that can cause a UCA have been identified, identify how the process models might occur due to received feedback or other information (or lack thereof).

[10] Tuning-related issues in control theory can be captured here. Process models may include controller beliefs about dynamic characteristics of a controlled process, and tuning problems can occur when those beliefs are incorrect. For example, proportional, integral, and derivative terms in PID controller represent beliefs about the dynamics of the controlled process. If those beliefs are incorrect, stability or other issues may occur and the control algorithm may produce unsafe control actions as a result.

Example:

UCA-2: BSCU Autobrake does not provide the Brake control action during landing roll when the BSCU is armed [H-4.1]

Controller process model (belief) that could cause the UCA: Controller believes the aircraft has already stopped on ground

Controller receives correct feedback but interprets it incorrectly: Wheel speed signals may momentarily reach zero during anti-skid operation, causing flawed process model

Scenario 1 for UCA-2: The BSCU is armed and the aircraft begins landing roll. The BSCU does not provide the Brake control action [UCA-2] because the BSCU incorrectly believes the aircraft has already come to a stop. This flawed process model will occur if the received feedback momentarily indicates zero speed during landing roll. The received feedback may momentarily indicate zero speed during anti-skid operation, even though the aircraft is not stopped.

Controller process model (belief) that could cause the UCA: Aircraft is in flight

Controller does not receive information when needed: Touchdown indication is not received

Scenario 2 for UCA-2: The BSCU is armed and the aircraft begins landing roll. The BSCU does not provide the Brake control action [UCA-2] because the BSCU incorrectly believes the aircraft is in the air and has not touched down. This flawed process model will occur if the touchdown indication is not received upon touchdown. <Why? This scenario needs to be finished.>

Any scenarios that involve inadequate feedback/information must be completed to explain why the feedback/information might be inadequate. Inadequate feedback and information cannot be prevented without understanding the reasons for it.

2) Causes of inadequate feedback and information

Whenever a scenario identifies feedback or information (or lack thereof) that can cause a UCA, we need to examine where the feedback/information comes from to explain what could cause those problems. Feedback comes from the controlled process (usually via sensors) and other information may come from other processes, other controllers, or other sources in the system or the environment.

In general, scenarios related to inadequate feedback and information might involve:

- Feedback or information not received
    - Feedback/info sent by sensor(s) but not received by controller
    - Feedback/info is not sent by sensor(s) but is received or applied to sensor(s)
    - Feedback/info is not received or applied to sensor(s)
    - Feedback/info does not exist in control structure or sensor(s) do not exist
- Inadequate feedback is received
    - Sensor(s) respond adequately but controller receives inadequate feedback/info
    - Sensor(s) respond inadequately to feedback/info that is received or applied to sensor(s)
    - Sensor(s) are not capable or not designed to provide necessary feedback/info

Scenarios involving feedback/info that is sent but not received or received inadequately may be caused by transmission errors, lost communication, delays in communication (including feedback/info sent but received in a different order), and other problems. Scenarios with an inadequate sensor response or no response may be caused by sensor failures, loss of power to the sensor, inaccuracies in sensor operation or measurement, sensor errors or misbehaviors, delays in sensor response, incorrect

configuration, degradation or changes to the sensor over time, unanticipated conditions in the sensor environment, or other problems. In addition, sensors may not be capable of providing the necessary feedback due to design errors, specification flaws, incorrect assumptions about the controlled process, sensors measuring the wrong conditions, sensors that report correct but misleading information (such as zero wheel speed when wheels are locked but the aircraft is moving), or other problems.

We can finish scenarios that involve feedback/info by identifying why that feedback/info might be received given the actual true state of the system (specified in the UCA context).

For example, let's finish Scenario 2 for UCA-2 above:

True state from UCA context: Aircraft is in landing roll (See Scenario 2 above)

Information received: Touchdown indication is not received upon touchdown (see Scenario 2 above)

How this could happen given the true state: Reported wheel speed is insufficient, reported weight on wheels is insufficient, wheel speed or weight on wheels indications are delayed, etc.

Scenario 2 for UCA-2: The BSCU is armed and the aircraft begins landing roll. The BSCU does not provide the Brake control action [UCA-2] because the BSCU incorrectly believes the aircraft is in the air and has not touched down. This flawed process model will occur if the touchdown indication is not received upon touchdown. The touchdown indication may not be received when needed if any of the following occur:

- Wheels hydroplane due to a wet runway (insufficient wheel speed)
- Wheel speed feedback is delayed due to filtering used
- Conflicting air/ground indications due to crosswind landing
- Failure of wheel speed sensors
- Failure of air/ground switches
- Etc.

As a result, insufficient deceleration may be provided upon landing [H-4.1]

To include causes related to security, only one additional possibility needs to be considered here: identify how the specified feedback and other information could be affected by an adversary. More specifically: how could they be injected, spoofed, tampered, intercepted, or disclosed to an adversary? For example, the following causes might be added to Scenario 2 above if security is included:

- Adversary spoofs feedback indicating insufficient wheel speed
- Wheel speed feedback is delayed due to adversary performing DoS attack
- Correct wheel speed feedback is intercepted and blocked by an adversary
- Adversary disables power to the wheel speed sensors

b) Identifying scenarios in which control actions are improperly executed or not executed

Hazards can be caused by UCAs, but they can also be caused without a UCA if control actions are improperly executed or not executed. To create these scenarios, we must consider factors that affect the control path as well as factors that affect the controlled process, as shown in Figure 2.19.

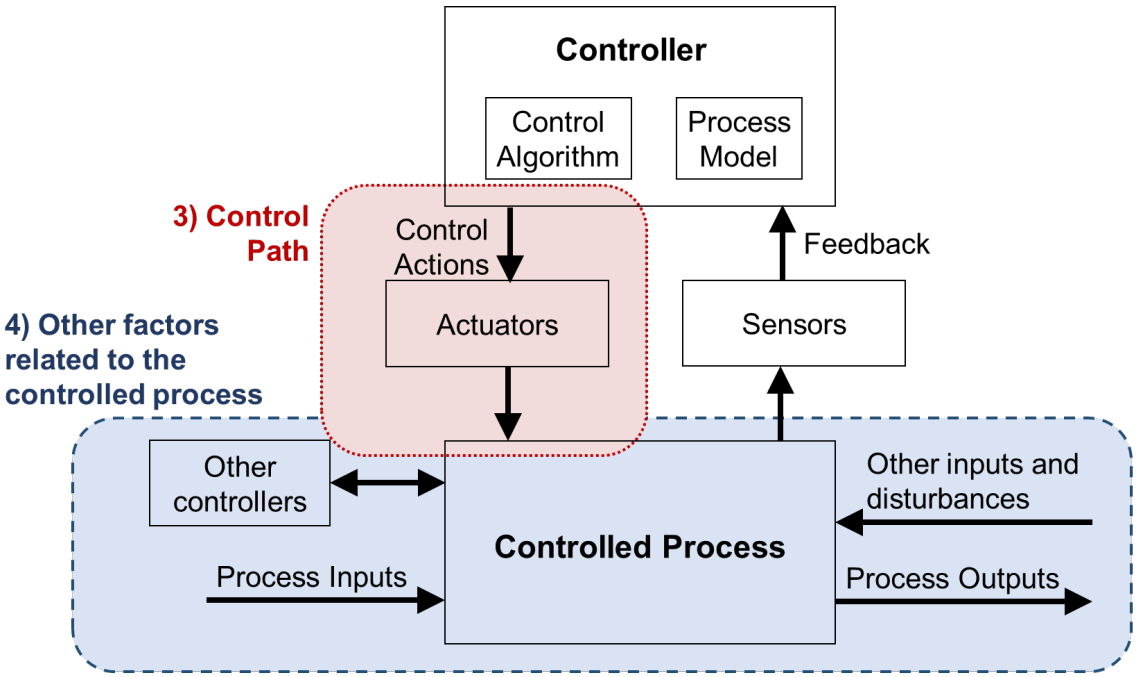


Figure 2.19: Generic control loop illustrating 1) the control path and 2) other factors that can affect the controlled process

1) Scenarios involving the control path

The control path transfers control actions to the controlled process. The control path may consist of a simple actuator, may involve a series of actuators, or it may transfer control actions through a complex network with switches, routers, satellites, or other equipment. Regardless of how it is implemented, we must identify how problems along this path can cause control actions to be improperly executed or not executed.

In general, scenarios involving the control path might include:

- Control action not executed
  - Control action is sent by controller but not received by actuator(s)
  - Control action is received by actuator(s) but actuator(s) do not respond
  - Actuator(s) responds but the control action is not applied to or received by the controlled process
- Control action improperly executed
  - Control action is sent by controller but received improperly by actuator(s)
  - Control action is received correctly by actuator(s) but actuator(s) respond inadequately
  - Actuator(s) respond adequately, but the control action is applied or received improperly at the controlled process
  - Control action is not sent by controller, but actuators or other elements respond as if it had been sent

Scenarios with control actions that are sent but improperly received or not received may be caused by delays in communication (including control actions sent but received in a different order), transmission errors, lost communication, and other problems. Scenarios with an improper actuator response or no response may be caused by actuator failures, loss of power to the actuator, inaccuracies in actuator operation, actuator errors or misbehaviors, delays in actuator response, other commands received by the actuator (including potentially conflicting commands from other controllers), incorrect

priority scheme used by the actuator, incorrect configuration, degradation or changes to the actuator over time, unanticipated conditions in the actuator environment, or other problems.

To create these scenarios, start with a control action, identify what improper execution or no execution means for your application, and identify how the control path could contribute to that behavior.

Example:

Control action: BSCU sends Brake command

No execution: Brakes not applied

Improper execution: Insufficient braking applied

Scenario 1: The BSCU sends the Brake command upon landing, but the brakes are not applied due to actuator failure. As a result, insufficient deceleration may be provided upon landing [H-4.1]

Scenario 2: The BSCU sends the Brake command upon landing, but insufficient braking is applied due to slow actuator response. As a result, insufficient deceleration may be provided upon landing [H-4.1]

Scenario 3: The BSCU sends the Brake command upon landing, but it is not received by the actuator due to a wiring error. As a result, insufficient deceleration may be provided upon landing [H-4.1]

It is also important to consider how control actions may not be sent but actuators or other elements respond as if a control action was sent. These scenarios will be similar to UCAs for control actions that are provided.

Example:

Control action: BSCU does not send Brake command

Improper execution: Brakes are applied during normal takeoff (similar to UCA-2)

Scenario 4: The BSCU does not send Brake command, but the brakes are applied due to hydraulic valve failure. As a result, acceleration may be insufficient during takeoff [H-4.6]

To include security problems, one additional possibility needs to be considered here: identify how the specified control action could be affected by an adversary. More specifically: how could it be injected, spoofed, tampered, intercepted, or disclosed to an adversary? For example:

Scenario 5: The BSCU sends the Brake command, but the brakes are not applied because an adversary executes a denial of service attack that blocks the Brake command. As a result, insufficient deceleration may be provided upon landing [H-4.1]

2) Scenarios related to the controlled process

Even if control actions are transferred or applied to the controlled process, they may not be effective or they may be overridden by other controllers.

In general, scenarios related to the controlled process might include:

- Control action not executed
    - Control action is applied or received by the controlled process but the controlled process does not respond
- Control action improperly executed
    - Control action is applied or received by the controlled process but the controlled process responds improperly

- Control action is not applied or received by the controlled process but the process responds as if the control action had been applied or received

These scenarios may be caused by process inputs that are missing or inadequate (such as inadequate hydraulic pressure, etc.), external or environmental disturbances, component failures, delayed response by the process, errors or misbehaviors in the process, potentially conflicting commands received from other controllers, previous control actions that were received or applied to the controlled process, incorrect priority scheme used by the process, incorrect configuration, degradation or changes to the process or the environment over time, unanticipated or unhandled conditions in the process environment, or other problems.

To create these scenarios, select a control action and identify what factors can affect the controlled process to make the control action ineffective.

Example:

> Control action: BSCU sends Brake command
>
> Scenario 6: The BSCU sends Brake command, but the brakes are not applied because the wheel braking system was previously commanded into alternate braking mode (bypassing the BSCU). As a result, insufficient deceleration may be provided upon landing [H-4.1]
>
> Scenario 7: The BSCU sends Brake command, but the brakes are not applied due to insufficient hydraulic pressure (pump failure, hydraulic leak, etc.). As a result, insufficient deceleration may be provided upon landing [H-4.1]
>
> Scenario 8: The BSCU sends Brake command, the brakes are applied, but the aircraft does not decelerate due to a wet runway (wheels hydroplane). As a result, insufficient deceleration may be provided upon landing [H-4.1]

To include security issues, the same additional possibility needs to be considered again: identify how adversaries can interact with the control process to cause the same issues. For example:

> Scenario 9: The BSCU sends Brake command, but the brakes are not applied because an adversary injected a command that put the wheel braking system into alternate braking mode. As a result, insufficient deceleration may be provided upon landing [H-4.1]

Tips to prevent common mistakes

The most common mistake is to identify individual causal factors rather than a scenario. For example, you may be tempted to create list of factors like “wheel speed sensor failure”, “wheel speed feedback is delayed”, “loss of power”, etc. The problem with listing individual factors outside the context of a scenario is that it’s easy to overlook how several factors interact with each other, you can overlook non-trivial and non-obvious factors that indirectly lead to UCAs and hazards, and you may not consider how combinations of factors can lead to a hazard. Considering single factors essentially reduces to a FMEA where only single component failures are considered.

Inputs and Outputs for Identifying Scenarios

Figure 2.20 summarizes the inputs and outputs for the scenario identification step.

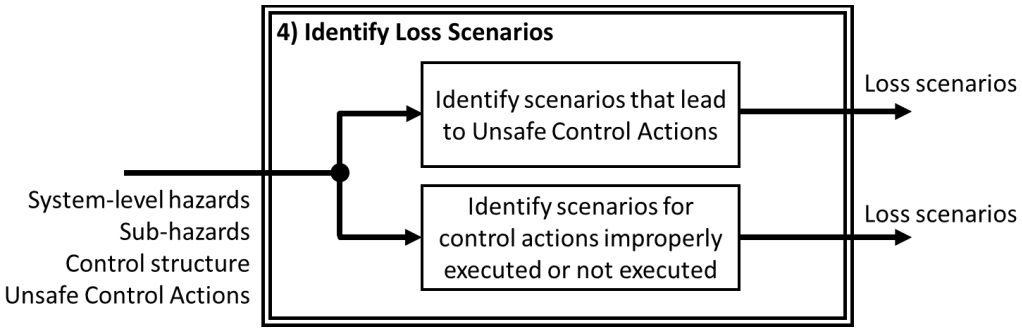


Figure 2.20: Overview of scenario identification

STPA Outputs and Traceability

Figure 2.21 shows the traceability that is maintained between various STPA outputs.

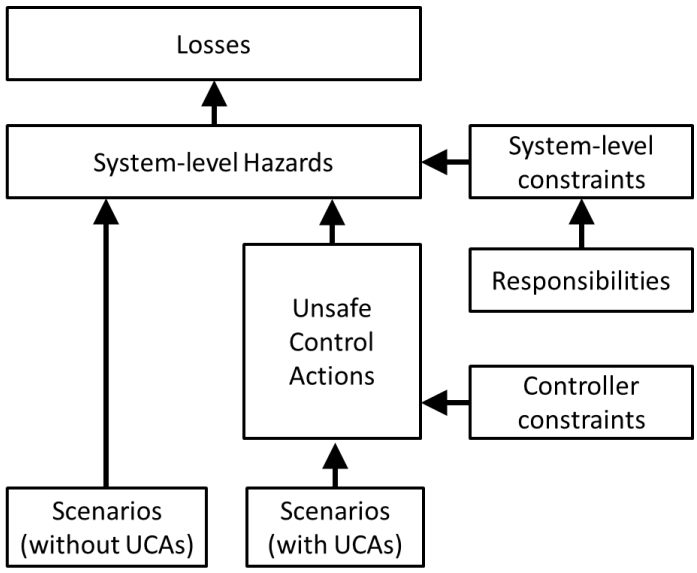


Figure 2.21: Traceability between STPA outputs

The control structure is closely related to every STPA output and therefore it is not explicitly shown in Figure 2.21 for simplicity. These STPA outputs can be used in many different ways, including:

- Drive the system architecture
- Create requirements
- Identify design recommendations
- Identify mitigations and safeguards needed
- Define test cases and create test plans
- Drive new design decisions (if STPA is used during development)
- Evaluate existing design decisions and identify gaps and changes needed (if STPA is used after the design is finished)
- Develop leading indicators of risk
- Design more effective safety management systems

- Etc.

You should now have a basic understanding of how STPA is performed. We suggest that you try the analysis on some of your own engineering problems. The Appendices provide some additional examples that may be useful.

Summary and a Look Forward

This chapter has described the basic approach used in STPA. The next chapter explains how STPA can be applied to the various phases and activities involved in system engineering. Chapter 4 provides additional information and examples for using STPA in workplace safety. Up to this point, the examples all involve technical systems but STPA can be used for any sociotechnical system. Chapter 5 shows how to use STPA in organizational analysis and for emergent, system-wide properties other than safety. Chapter 6 shows how STPA can be used to identify leading indicators of risk. Chapter 7 describes the use of STAMP and STPA to design more effective safety management systems. Finally, Chapter 8 describes what we have learned so far about how to integrate STPA into large organizations and projects.

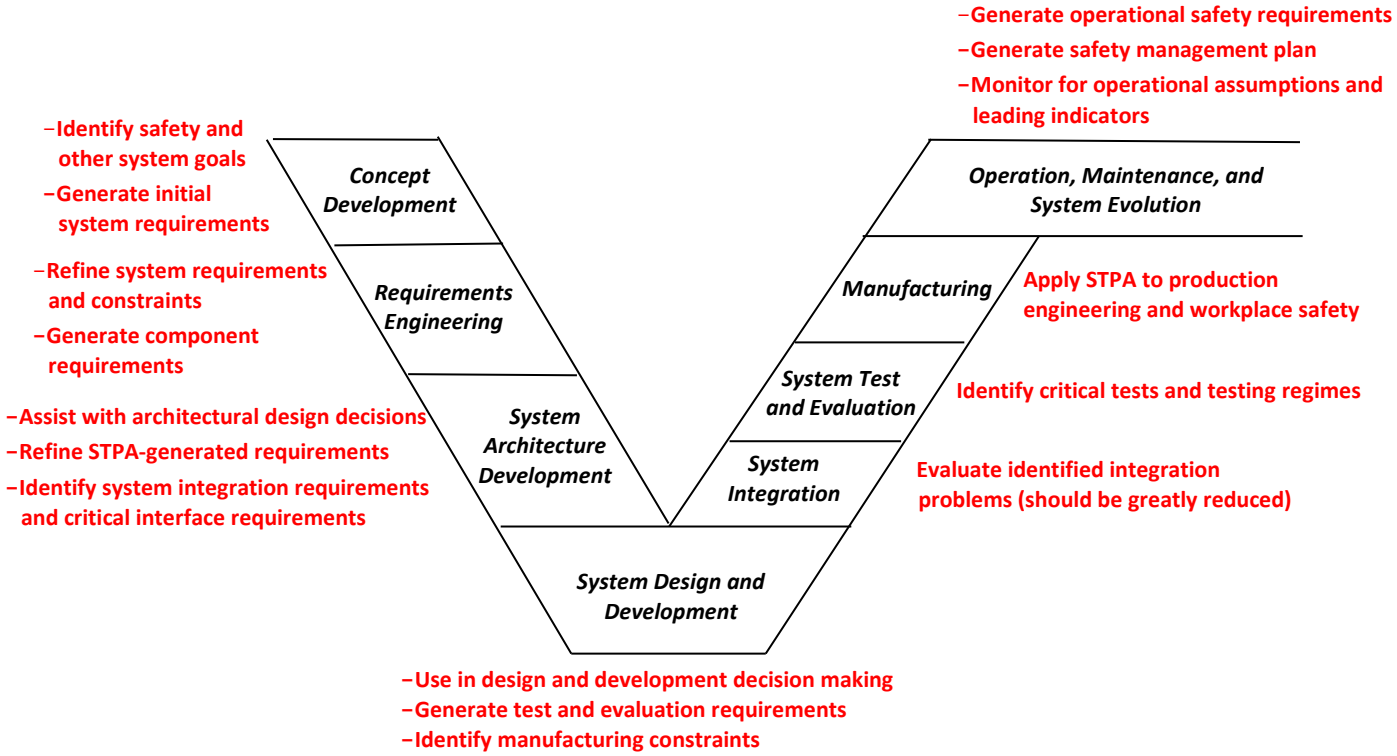
Chapter 3: Integrating STPA into the System Engineering Process

Nancy Leveson

Too often, system safety is isolated or separated in some way from the system engineering process. The most common result is that safety is treated as an after-the-fact assurance activity. Because safety cannot be assured into a system but must be designed in, safety-related design flaws are often found late, when they cannot be fixed. At that point, the effort then focuses on trying to find arguments that the identified flaws do not need to be fixed. When those arguments cannot be sustained, the efforts to deal with the safety flaws often devolve to making expensive and not very effective solutions, such as redundancy or expecting the operators of the system to detect and fix problems through far-from-ideal procedural solutions.

This chapter of the handbook describes how to tightly integrate system safety analysis into the entire system engineering process (see Appendix F for a basic introduction to system engineering). The result will be a significant decrease in the cost of engineering for safety as well as greatly increased effectiveness and, hopefully, fewer losses. It can also reduce rework, which reduces cost and schedule.

The figure below shows a simplified version of the standard system engineering V-model (or waterfall model drawn with a kink in it). The feedback loops are omitted simply to declutter the diagram. This figure is used to illustrate how to integrate STPA into the standard system engineering process. The potential roles for STPA are shown in red. If you use a variant of the V-model or a different development model, it should not be difficult to translate from the standard V-model to most any other model except, perhaps, for those that omit the early processes (the upper two activities on the left leg of the V). In that case, you should not be building safety-critical systems.



STPA can be used throughout the standard system engineering process, starting in the earliest concept development stage. In essence, STPA can be used to generate high-level safety requirements early in the concept development phase, refine them in the system requirements development phase. The system requirements and constraints can then assist in the design of the system architecture and more detailed system design and development. The STPA result process then goes hand in hand with design and development as the analysis can be used to inform decisions as they are made. STPA continues to be useful through assurance and manufacturing and provides important information for use during operations

STPA fits into a model-based engineering process as it works on a model of the system (which is refined as design decisions are made) although that model is different than the architectural models usually proposed for model-based system engineering today. STPA promotes traceability throughout the development process so decisions and designs can be changed with minimum requirements for redoing previous analyses.

Finally, as noted in many places in this handbook, STPA can be applied to any emergent system property in the system engineering and product lifecycle, not just safety.

Overall Process

STAMP and STPA contribute to all the activities in system engineering. The use of the analysis and its results for the following activities are described in the rest of this chapter:

1. Definition of losses to be handled during development and operation
2. Identification of external constraints (including market and regulatory requirements) on the system design
3. Identification of system-level hazards and the related requirements and constraints on system behavior
4. Modeling of the system control structure
5. Refinement of hazards and constraints and allocation of functions to system components
6. Assist with making architectural, design, and implementation decisions
7. System Integration assistance
8. Generation of system test requirements
9. Control of manufacturing (production engineering, workplace safety)
10. Generation of operational safety requirements (including leading indicators of increasing risk) and the safety management plan
11. Operational safety management including monitoring leading indicators

1. Decisions about Losses to be Considered

While early Concept Development may differ in specific variants of the V-model, this stage usually includes such activities as stakeholder and user analysis (needs analysis), customer requirements generation, regulatory requirements review, feasibility studies, concept and trade space exploration, and establishment of criteria for evaluation of the evolving and final design. Toward the end of this step, there may be creation of a Concept of Operations.

Too often, this early stage of system engineering is not given the attention and effort it deserves and development proceeds almost immediately with system architecture specification and high-level design. Inadequate concept development may, however, lead to systems that are not usable by the customer, only partially satisfy stakeholder needs, or are difficult to assure, maintain, and operate. While changes

may be made later in development to make up for omissions in the early concept development stage, these later changes are increasingly expensive and disruptive as development proceeds.

Safety and security concerns, in particular, are often inadequately considered in early concept exploration. Figure 3.1 shows a typical approach to handling these emergent system properties [Young 2017]. Major emphasis is often placed on responding to a loss only after it occurs in operations. In addition, the major focus may be on adding "bolt-ons" (e.g., protection systems or intruder detection) after the bulk of design and system engineering has already been done. Making changes like these late in the development process is not only more expensive, the fixes are usually much less effective than if safety and security had been built into the system from the beginning.

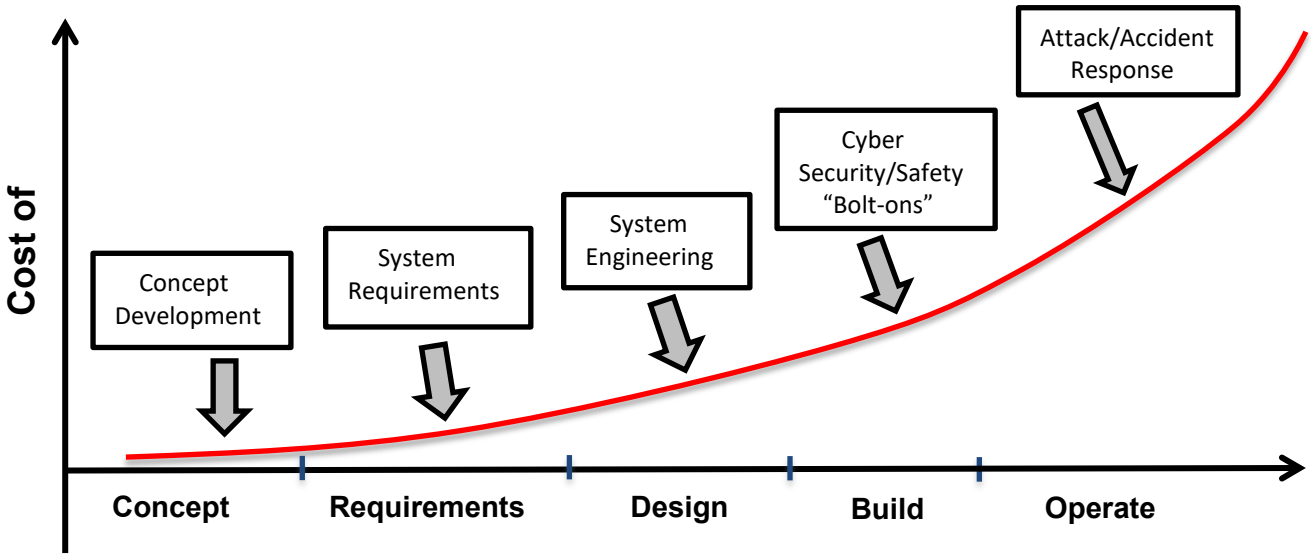


Figure 3.1: Safety and security need to be built into the system from the beginning of development[11]

Instead, safety and security considerations should be initiated in the concept development stage and the results used to generate the safety and security requirements for the overall system and its components. Frola and Miller (1984)[12] claimed that 70-90% of the design decisions related to safety are made in the concept development stage and changing these decisions later may be infeasible or enormously expensive. The same is true for security.

Safety may, in many system engineering processes, be considered at this stage, but usually in the form of a Preliminary Hazard Analysis, which attempts to provide a risk assessment to determine how much effort should be put into the identified hazards during development and operations and some preliminary recommendations on how to eliminate or control them.

[11] William Young, *A System-Theoretic Security Analysis Methodology for Assuring Complex Operations Against Cyber Disruptions*, Ph.D. dissertation, MIT, 2017.

[12] F. Ronald Frola and C.O. Miller, System Safety in Aircraft Acquisition, Technical Report, Logistics Management Institute, Washington D.C., January 1984.

Table 2.1. A typical PHA format[13]

PROGRAM: _____ DATE: _____ ENGINEER: _____ PAGE: _____						
ITEM	HAZARD COND	CAUSE	EFFECTS	RAC	ASSESS-MENTS	RECOMM-ENDATIONS
Assigned number	List the nature of the condition	Describe what is causing the stated condition to exist	If allowed to go uncorrected, what will be the effect or effects of the hazardous condition	Hazard Level assign-ment	Probability, possibility of occurrence: -Likelihood -Exposure -Magnitude	Recommended actions to eliminate or control the hazard

[Vincoli, 2005]

While most of the information in the table is available in the early concept development stage, the probability or likelihood of the hazardous condition cannot be known before any detailed design has been done or even after that time if the system is software-intensive. Historical information is not useful when the system differs significantly from past systems. We have found using STPA that specific hazards on real projects that have perfectly reasonable causal scenarios are often incorrectly dismissed early in the system development process as "marginal" or extremely unlikely. Research by psychologists has shown that humans are very bad at estimating probability or likelihood of unusual events.[14]

The biggest problem is that this type of PHA does not provide the information necessary to identify the functional safety and security requirements for the system, which is the primary goal in early concept analysis. Traditional hazard analysis techniques, which theoretically might be used for this purpose, require a fairly detailed design and thus are not appropriate until much later in the development process, when it is too late to provide the functional safety requirements for the design effort.

An alternative often used in the aviation community is to generate probabilistic requirements to be satisfied by the designed system [SAE ARP 4761]. Examples include "Loss of all wheel braking during landing or rejected take off shall be less than 5E-7 per flight" and "Undetected inadvertent wheel braking on one wheel without locking during takeoff shall be less than 5E-9 per flight" [SAE ARP 4761]. While these types of requirements might have been reasonable in the era when braking systems were composed almost exclusively of hardware parts with known failure probabilities, the extensive use of software in almost all systems make it impossible to ensure that these requirements are satisfied in the software-intensive systems being engineered today.

STAMP and STPA provide the ability to generate the information needed in the concept development stage—identifying stakeholder and user needs (needs analysis), generating customer goals and requirements, reviewing regulatory requirements, evaluating feasibility, exploring system concepts and the conceptual trade space, and establishing criteria for evaluation of the evolving and final design. The

[13] Jeffrey Vincoli, Basic Guide to System Safety, John Wiley & Sons, 2005.

[14] See, for example, the work of Tversky and Kahneman on heuristic biases.

primary problems involved in many failed or problematic system engineering efforts stem from limited understanding of the stakeholder needs and the appropriate system goals.

2. Identification of external constraints (including market and regulatory requirements) on the system design

The social system safety control structure can be used to understand the external (environmental) social constraints on the system. Figure 3.2 shows an example sociotechnical control structure that is similar to that common to many regulated industries. There is both a control structure for development (on the left) and for operations on the right. Each industry will probably have its own sociotechnical control structure, and it may differ by country. Only an example national structure is shown here but in some industries (e.g., aviation and nuclear) there are overlying international control components.

Once you have modeled this structure, there will probably be few changes for each development project undertaken but some differences may exist. The structure can be used to identify and document regulatory and other external requirements for the system being developed.

3. Identification of Losses to be Considered

At the very early conceptual development stage, stakeholders need to specify unacceptable losses. While many hazard analysis and system safety techniques handle only human death or injury and property damage, STPA can handle any loss including environmental pollution, mission loss, monetary losses, and even damage to company reputation. In other words, the losses represent any emergent system property that the stakeholders want to avoid. Formally, an accident (or mishap, which is the term used by the military) is any undesired or unplanned event or condition that results in what stakeholders consider to be a loss. The aircraft example that has been used in this handbook starts from the following accidents (losses):

A1. Death or serious injury to aircraft passengers or people in the area of the aircraft

A2. Unacceptable damage to the aircraft or objects outside the aircraft

What is considered to be “unacceptable” must be defined by the stakeholders. Additional losses might be specified if they are considered to be worth putting in effort to avoid them such as:

A3: Financial losses resulting from delayed operations

A4: Reduced sales due to damage to aircraft or airline reputation.

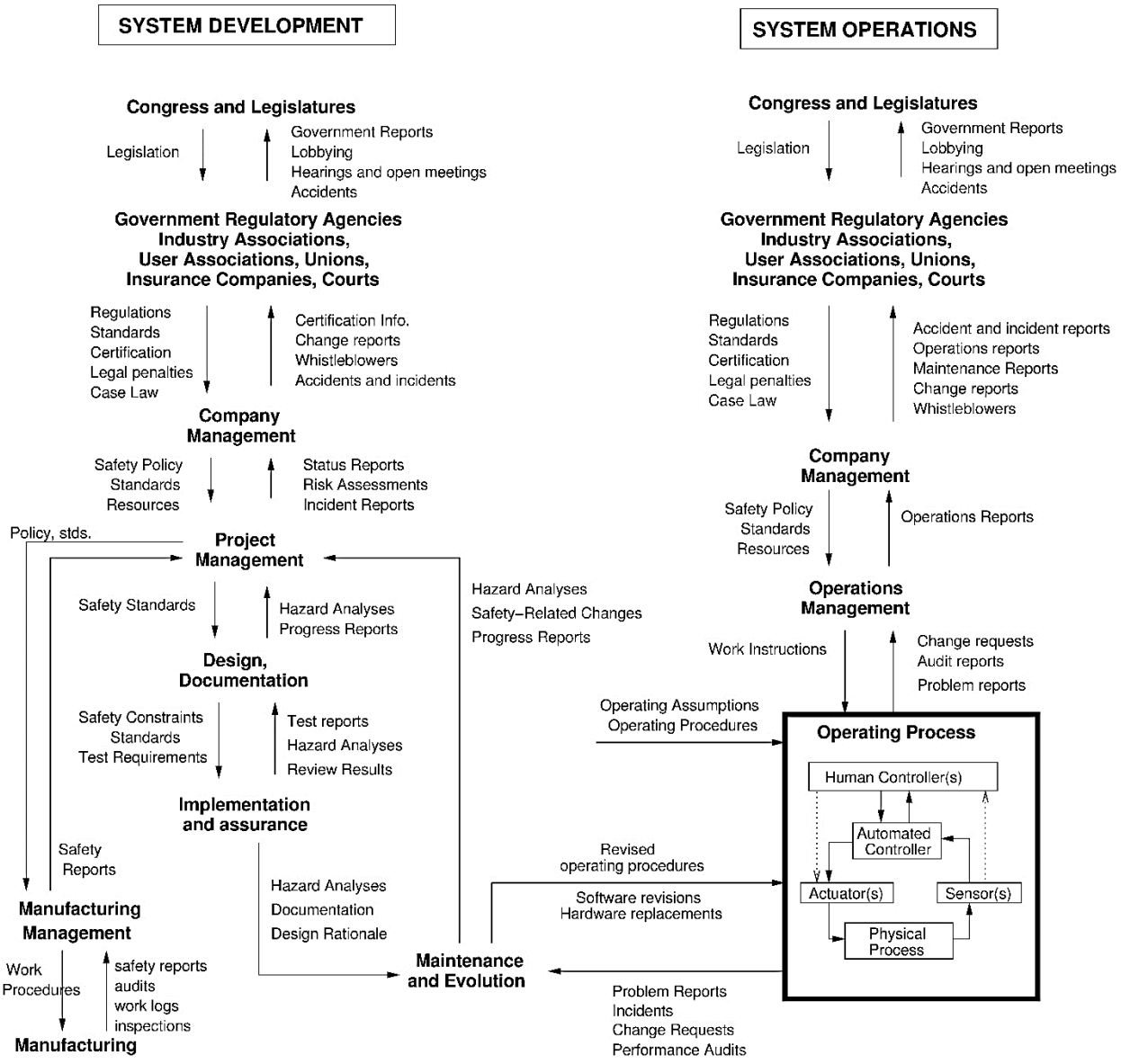


Figure 3.2: An Example Safety Control Structure

4. Identification of system-level hazards and the related constraints on system behavior

Once unacceptable losses are identified by the stakeholders, engineers can define the hazards that are associated with those losses. The ultimate goal is to eliminate or, if not possible, to control or reduce identified hazards during the system design activity.

The first problem is how to define the hazards. Although the term hazard is sometimes used to refer to things outside the system boundaries, such as inclement weather in aviation, hazards in STPA are limited to system states that are within the control of the system designer. For example, the hazard is not the inclement weather but rather it is the aircraft being negatively impacted by inclement weather. Constraints or controls may include staying clear of the weather or designing the aircraft to withstand

the impact of the weather. Because system engineering is concerned with the design of the system, it can have an impact only on the behavior of the system itself and cannot change things outside the system boundary (i.e., in the system operating environment).

Sometimes a hazard is defined as a condition that could lead to an accident or a precondition for an accident. But that definition includes almost everything, most of which cannot be eliminated or controlled if the system is to operate at all. For example, an aircraft that is landing could lead to an accident later on, but we cannot build a useful system where aircraft never land. As another example, the hazard is not deceleration on landing (which is required for safe operations) but inability to decelerate after landing.

So for practical reasons, hazards are defined as precursor states to an accident that the system designers never want to occur and thus should be eliminated or controlled in the design process. A formal definition was provided in the previous chapter.

Clearly, in order to eliminate the hazards in the system design, they need to be identified at the beginning of the system development process. The specified hazards are used to create the behavioral (functional but not probabilistic) safety requirements for the various system components, including the software and human operators.

For the accidents A1 and A2 above, the hazards include**:**

H1: Insufficient thrust to maintain controlled flight [A1, A2]

H2: Loss of airframe integrity [A1, A2]

H3: Violating minimum separation between aircraft and fixed or moving objects [A1, A2]

H4: An aircraft on the ground comes too close to moving or stationary objects or inadvertently leaves the taxiway [A1, A2]

H5: Aircraft unable to take off when scheduled [A3, A4]

H6: etc.

These hazards lead to very high-level system constraints[15] such as:

SC1: Sufficient thrust must be available to maintain controlled flight [H1]

SC2: Airframe integrity must be maintained under worst case conditions [H2]

SC3: Aircraft must satisfy minimum separation standards from fixed or moving objects [H3]

SC4: Aircraft on the ground must always maintain a safe distance from moving or stationary and objects and remain within safe regions such as taxiways.

SC5: Aircraft must be able to take off within TBD minutes of scheduled departure [H6]

SC6: etc.

5. Modeling the Functional System Control Structure

While the first control structure model developed can include the environment in which the system will operate (outside the system boundary), the system control structure in this step provides more detail about what is within the system design space. The scope of the model (which is the definition of

[15] Because negative requirements are often not allowed and changing safety constraints into positive statements is sometimes not possible without losing important information, requirements and constraints are distinguished here. Constraints are described using "must" or "must not" rather than "shall." There is an additional advantage in that a distinction is made between requirements (system goals or mission) and constraints on how those goals can be achieved. The differentiation is useful when tradeoff decisions need to be made.

the boundaries of the system to be considered) will depend on the goal of the system engineering process. Multiple different models may be useful, with increasing detail being provided as the conceptual design process proceeds. Details on building control structure models are provided in the previous chapter. Some repetition is included here to make the chapters free-standing.

Looking at the control structure in Figure 3.2, the system to be designed might be the functional system control structure within the lower right-hand box of the figure labeled “Operating Process.” STPA is usually first performed on this system control structure to identify the very high-level system requirements and constraints. During the conceptual design process, the control structure will be refined and STPA is used to refine the associated system requirements and constraints.

For example, Figure 3.3 shows a very high-level functional control model of an aircraft, with just three components: the pilot, the automated control system (which will probably consist of multiple computers, but that decision is left for a later design stage), and the physical aircraft components. For complex systems, such as aircraft, levels of abstraction can be used to zoom in on the pieces of the control structure currently being considered. This type of top-down refinement is also helpful in understanding the overall operation of the aircraft and in identifying interactions among the components.

Note that there are no design decisions in this model beyond the fact that the system will be an aircraft and the aircraft will include the general functions and design of any aircraft. Therefore, it is appropriate for use at the early concept development stage. If any of the parts of the model are too detailed for a particular application at this point in system engineering, then they can be removed. Note that the pilot may be a human (onboard or on the ground) or automation (for unmanned aircraft) and the aircraft could even be fixed wing, vertical lift, or VTOL (vertical takeoff and landing). No distinction is necessary at this point of conceptual development. This generality allows decisions to be put off until the more information is available and also allows reuse of the models and the analysis. Aircraft systems being designed today may include several or all of these types of piloting configurations.

The role of the pilot, as shown in the Figure 3.3 control structure, is to manage the aircraft automation and, depending on the design of the aircraft, directly or indirectly control takeoff, flight, landing, and maneuvering the aircraft on the ground. The pilot and the automated controllers contain a model of the system or subsystem that they are controlling. The automation is controlling the aircraft so it must contain a model of the current aircraft state. The pilots also need a model of the aircraft state, but in addition they need a model of the state of the automation and a model of the airport or airspace in which they are operating. Many pilot errors can be traced to flaws in their understanding of how the automation works or in their understanding of the current state of the automation.

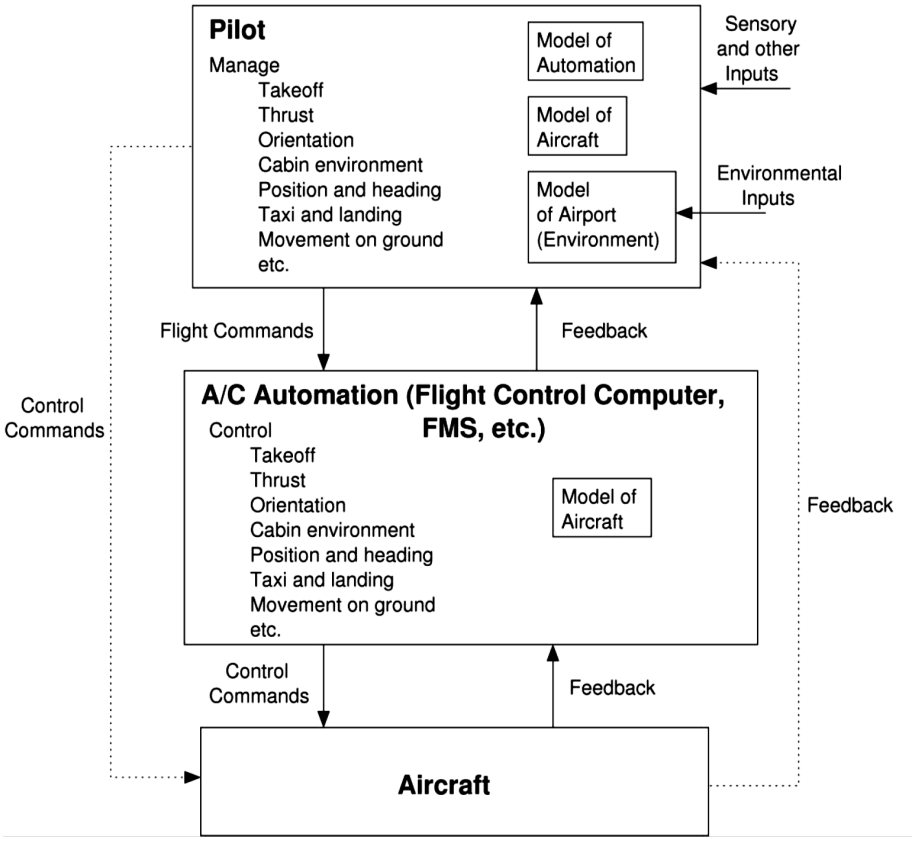


Figure 3.3: A High-Level Control Structure at the Aircraft Level

Pilots provide flight commands to the automation and receive feedback about the state of the automation and the aircraft. In some designs, the pilot can provide direct control actions to the aircraft hardware (i.e., not going through the automated system) and receive direct feedback. The dotted lines represent this direct control and feedback. As the design is refined and more detailed design decisions are made, these dotted line links may be eliminated or instantiated with specific content. The pilot always has *some* direct sensory feedback about the state of the aircraft and the environment (unless the pilot is on the ground, as in UAS) through various sensory modalities. At the same time, pilots often have limited direct sensory feedback about the state of the automation and the physical components the automaton is controlling; the information available to them will be limited to those items the engineers deemed important for them to know. STPA helps to identify what this critical information is. Some advanced flight controls today contain states and inputs that provide no feedback to the pilot at all because the engineers have assumed they do not need it or have omitted it for some other reason.

STPA is done using this control structure to refine the very general system-level safety requirements and constraints (examples shown above) developed from the system hazards. These new refined safety requirements will specify the responsibilities, authority, and accountability of the three components in the high-level control structure as well as the contents of the process models for each component and the feedback requirements in order to maintain safe operation of the aircraft. For example, one requirement may be that the pilot commands adequate thrust to ensure safe maneuvering of the aircraft at each stage of operation [**SC1**] and another is that the pilot shall provide safe commands to control movement on the ground [**SC4**]. Note that the system-level safety requirements are now starting to be allocated to individual components in the safety control structure.

6. Refinement of Hazards and Safety Constraints and Allocation to System Components

As the conceptual design process continues, the control structure will be refined as well as the requirements. This refinement process results from the identification of unsafe control actions and their causal scenarios appropriate to the current level of design detail. The result will be a set of safety-related requirements that can be refined and used for the rest of the development process.

The system and safety requirements should precede creation of the system architecture used to satisfy them. Of course, safety is not the only goal for any system and other requirements will need to be considered in the development of the architecture. But, in our experience, architectures are often developed before the safety (and often the system) requirements are identified. This inverted sequence raises the likelihood that the architecture will not be optimized for and sometimes not even appropriate for the system goals.

The architectural development will include the allocation of system requirements to the system components. This process will include, for STPA, generation of causal scenarios for the unsafe control actions generated at the system level. Deceleration is used as an example of this refinement process. The relevant high-level system hazard here is

H4: An aircraft on the ground comes too close to moving or stationary objects or inadvertently leaves the taxiway [A1, A2]

The specific accidents related to H4 occur when the aircraft operates on or near the ground and may involve the aircraft departing the runway or impacting object(s) on or near the runway. Such accidents may include hitting barriers, other aircraft, or other objects that lie on or beyond the end of the runway at a speed that causes unacceptable damage, injury or loss of life.

H4 can be refined into the following deceleration-related hazards:

H4-1: Inadequate aircraft deceleration upon landing, rejected takeoff, or taxiing

H4-2: Deceleration after the V1[16] point during takeoff

H4-3: Aircraft motion when the aircraft is parked

H4-4: Unintentional aircraft directional control (differential braking)

H4-5: Aircraft maneuvers out of safe regions (taxiways, runways, terminal gates, ramps, etc.)

H4-6: Main gear wheel rotation is not stopped when (continues after) the landing gear is retracted

The high-level system safety constraints associated with these hazards are a simple restatement of the hazards as constraints on the design.

SC1: Forward motion must be retarded within TBD seconds of a braking command upon landing, rejected takeoff, or taxiing (H4-1).

SC2: The aircraft must not decelerate after V1 absent direct pilot action (H4-2).

SC3: Uncommanded movement must not occur when the aircraft is parked (H4-3).

SC4: Differential braking must not lead to loss of or unintended aircraft directional control (H4-4)

[16] The V1 point is that point in the takeoff sequence where it is more dangerous to stop the takeoff than to continue. In some very rare circumstances, it is safer for the pilot to decelerate after the V1 point than to continue the takeoff procedure. These situations (and a few others) are ignored in this example analysis to allow the results to fit in a few pages.

SC5: Aircraft must not unintentionally maneuver out of safe regions (taxiways, runways, terminal gates and ramps, etc.) (H4-5)

SC6: Main gear rotation must stop when the gear is retracted (H4-6)

Note that all of these are very high-level, i.e., they start with consideration of the aircraft as a whole. As explained in the previous chapter, they will be refined through an iterative process into a more detailed set of functional safety requirements that are associated with specific system components, including the crew, the software, and the component interactions. System safety requirements are generated to prevent the causal scenarios identified by STPA. The process continues until the hazards have all been adequately analyzed and handled in the design. The important point is that the system behavior as a whole is considered first so that potential unsafe interactions among components can be identified at the beginning of the process without having to try to consider all interactions later and determine whether they are hazardous or not.

Figure 3.4 zooms in on the control structure model for the ground control function, which is where deceleration occurs. There are three basic physical components being controlled: the reverse thrusters, the spoilers, and the wheel braking system. Again, by including the larger functional control structure rather than simply one of these, STPA can consider all interactions (both intended and unintended) among the braking components related to the hazard being analyzed. There are clearly some design decisions shown here in that the braking system is determined to contain reverse thrusters, spoilers, and wheel brakes. This level of design detail, however, is still very general and high-level and satisfies most aircraft.

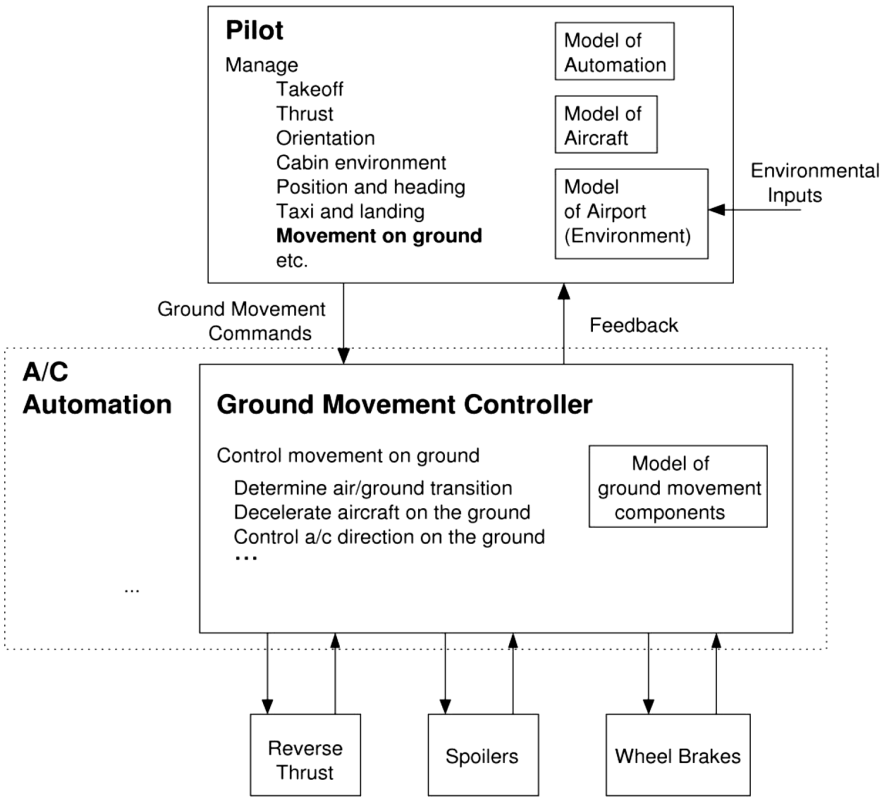


Figure 3.4: Deceleration Control Structure

Examples of safety constraints on the interaction between the braking system components that can be generated by STPA at this point in development include:

SC-BS-1: Spoilers must deploy when the wheel brakes are activated manually or automatically above TBD speed.

SC-BS-2: Wheel brakes must activate upon retraction of landing gear.

SC-BS-3: Activation of ground spoilers must activate armed automatic braking (autobrake) system.

SC-BS-4: Automatic braking system must not activate wheel brakes with forward thrust applied.

SC-BS-5: Automatic spoiler system must retract the spoilers when forward thrust is applied.

Note that because STPA includes humans, such as operators, maintainers, and even managers, STPA provides a structured method to identify human errors influenced by the system design, such as mode confusion and loss of situational awareness, that can lead to hazards. These human errors can arise due to a loss of consistency (lack of synchronization) between the actual automation state and the operator's mental model of that state.

Figure 3.5 further zooms into the deceleration control structure by focusing on the functional structure of the wheel brake system. Again, although only one component of the braking system is considered in Figure 3.5, there are no assumptions about the physical design and implementation. STPA starts without a specific design solution to potential problems. Instead, it starts from the basic required functional behavior and identifies the conditions under which that behavior can be hazardous. Designers can later decide on particular design solutions, such as redundancy or redesign of the functions, necessary to satisfy the safety requirements derived through this analysis.

Figure 3.5 does include one important design decision to include an autobraking function, which is common in modern aircraft. Pilots can preset an autobrake before they land in order to ensure consistent braking action is applied. Autobraking can reduce pilot workload under gusty or icy conditions and greatly reduce brake wear. If set, the autobrake will engage the wheel brakes at the specified time.

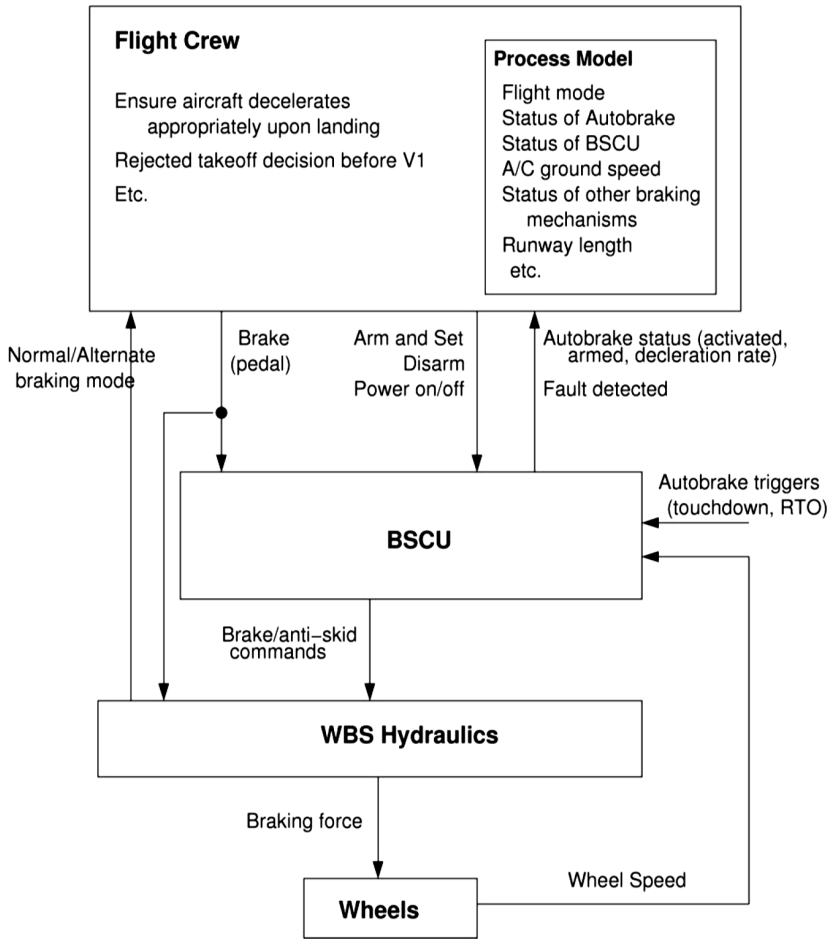


Figure 3.5: The Control Structure for the Wheel Braking System

Table 3.1 shows some of the safety constraints generated by STPA for Figure 2.6 for the flight crew (FC) and the Brake System Control Unit (BSCU). These are derived from the UCA Context Table generated as described in earlier chapters of this handbook. Rationale is included in Table 3.1 as is traceability information to assist in changing design decisions later without introducing unsafe features.

Table 3.1: Example STPA Generated System-Level Safety Constraints

Unsafe Control Action	Description	Rationale
FC-R1	Crew must not provide manual braking before touchdown [CREW.1c1]	Could cause wheel lockup, loss of control, or tire burst
FC-R2	Crew must not stop manual braking more than TBD seconds before safe taxi speed reached [CREW.1d1]	Could result in overspeed or runway overshoot
FC-R3	The crew must not power off the BSCU during autobraking [CREW.4b1]	Autobraking will be disarmed
BSCU-R1	A brake command must always be provided during RTO [BSCU.1a1]	Could result in not stopping within the available runway length

BSCU-R2	Braking must never be commanded before touchdown [BSCU.1c1]	Could result in tire burst, loss of control, injury, or other damage
BSCU-R3	Wheels must be locked after takeoff and before landing gear retraction [BSCU.1a4]	Could result in reduced handling margins from wheel rotation in flight

The high-level system architecture could be defined using the system requirements and these safety constraints at this point in development. The STPA-generated causal scenarios for violation of these high-level safety constraints will provide more detailed design constraints and other information to assist in the detailed architectural and system design process.

As design decisions are made, the STPA analysis is continually iterated and refined to assist the designer in making tradeoffs and effective design decisions. For the braking example, a more detailed control structure that shows some design decisions (including the valves used by the hydraulic controller) is shown in Figure 3.6. The design decisions, involving the additional of valves and commands related to the hydraulic controller might be at least partially the result of resolving the hazardous scenarios already identified.

Continuing the STPA analysis at this more detailed level of design involves the identification of unsafe control actions and the associated safety constraints related to the BSCU hydraulic controller (HC) when controlling the three individual valves included in the refined design (Table 3.2):

Table 3.2: Example STPA Generated System-Level Safety Constraints On the Wheel Brake System Control Structure

Unsafe Control Action	Description	Rationale
HC-R1	The HC must not open the green hydraulics shutoff valve when there is a fault requiring alternate braking [HC.1b1]	Both normal and alternate braking would be disabled.
HC-R2	The HC must pulse the anti-skid valve in the event of a skid [HC.2a1]	Anti-skid capability is needed to avoid skidding and to achieve full stop in wet or icy conditions.
HC-R3	The HC must not provide a position command that opens the green meter valve when no brake command has been received [HC.3b1]	Crew would be unaware that uncommanded braking was being applied.
HC-R4	The HC anti-skid must not release brakes when on low speed taxi.	Anti-skid system senses skid by sensing low wheel rotation or a rapid change in wheel rotation speed. This could lead to inadvertent release of brakes during low-speed taxi.

Once again, traceability is maintained with the unsafe control actions and scenarios from which these functional design constraints were derived. Those control actions and scenarios are in turn traceable back to higher levels of design and analysis. The result is an elaborate tree structure of a tracing of the STPA analysis throughout the design process that will assist in making changes or in understanding where certain constraints came from.

Additionally, more detailed causal scenarios associated with the unsafe control actions can be generated along with more design decisions about how to eliminate or mitigate the new scenarios. Alternative designs might be analyzed to provide information about how to resolve tradeoffs between alternative design choices. The process continues throughout the entire system development process.

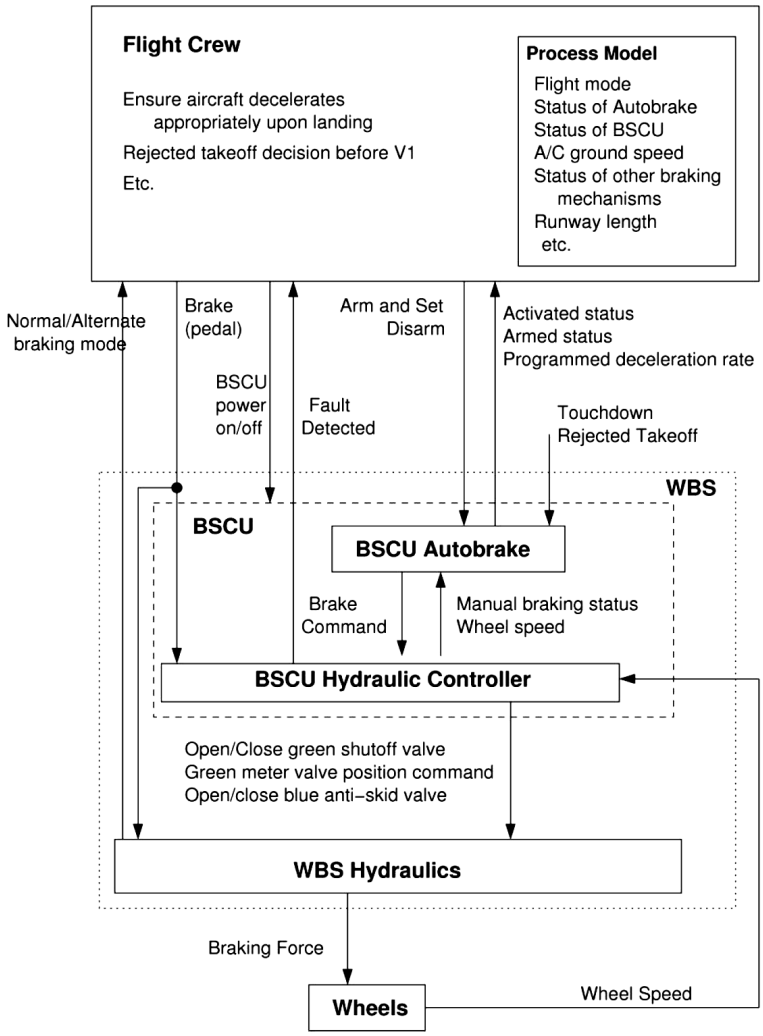


Figure 3.6: A More Detailed Version of the Wheel Brake System Control Structure

7. System Integration

System integration should be a lot less painful (at least from a safety standpoint) if STPA is used to identify the integrated system safety requirements early in the development process. Unsafe interactions among components, rather than being found now, should have been designed out of the system before this late development phase. Any safety-related integrated system flaws found at this point should be traced back to flaws in the development process (described above) and eliminated from future development projects. Of course, the flaws found will need to be incorporated into the STPA analysis to identify constraints to eliminate the flaws.

8. Generation of System Test and Evaluation Requirements

The causal scenarios and other information generated in the STPA analysis during the activities on the left-side of the "V" can be used to generate test requirements and evaluation programs. The test requirements and planning will depend on the type of system and testing being done.

As an example, Dan Montes, a U.S. Air Force test pilot, in his MIT Ph.D. dissertation, describes how STPA can be used in aircraft flight testing. Figure 3.7 shows a version of the example safety control structure in Figure 3.2 augmented with a separate testing control structure embedded between the development and operations control structures. Note that new hazards, unsafe control actions, and causal scenarios may exist in the flight test program. For example, new contexts may need to be considered for the test flight environment that might not exist in the normal operational environment (e.g., testing the results of abnormal stress on the aircraft or applying extreme control actions to test for recoverability). STPA can be applied to this system testing control structure, just as it is on any control structure, and the results used for flight test safety planning and in generating flight test plans.

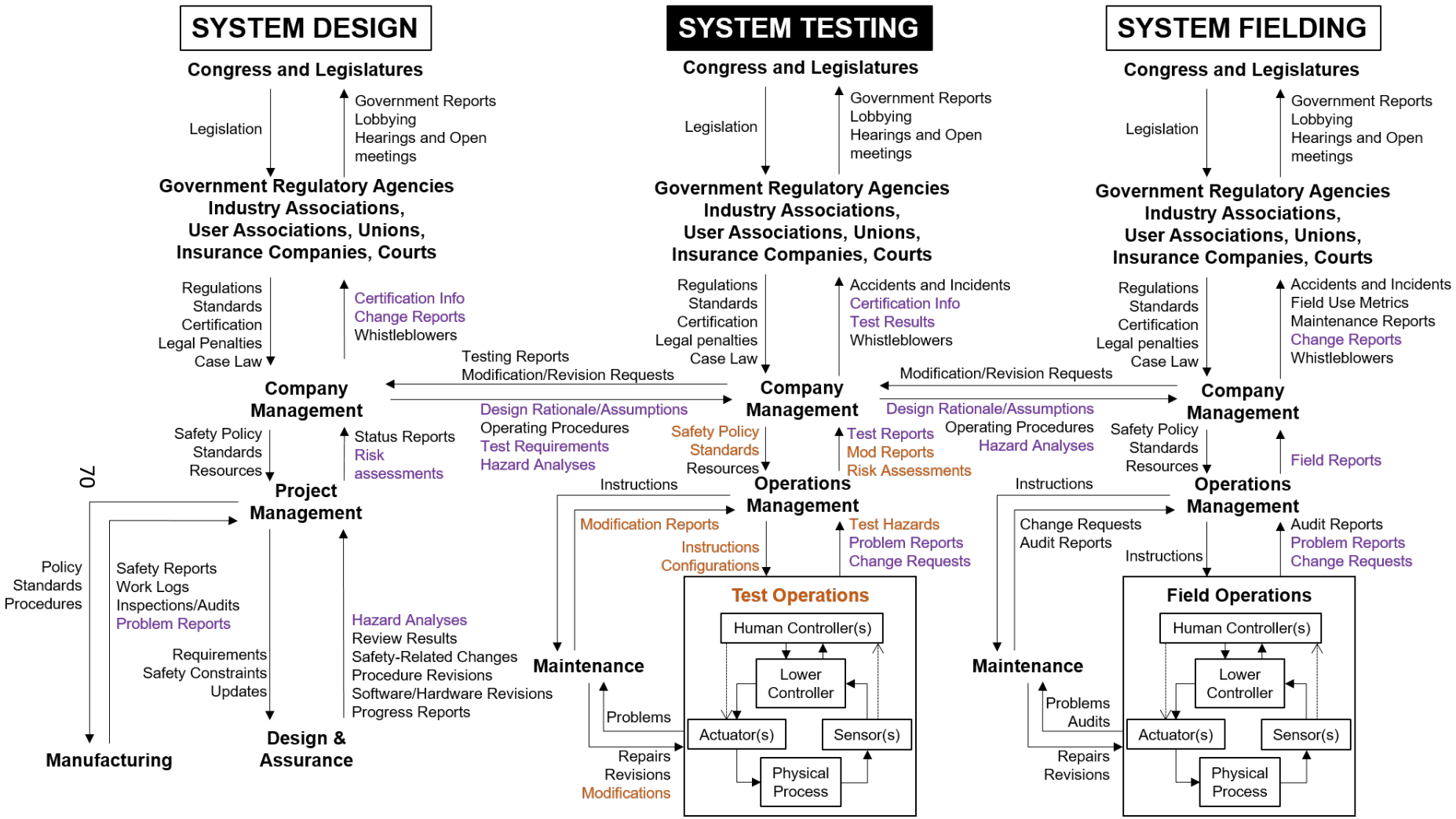
9. Control of Manufacturing (Production Engineering, Workplace Safety)

STPA has been applied successfully to both production engineering and to workplace safety. Considerations of producibility (particularly with respect to safety in manufacturing) should be included in the original STPA analysis if manufacturing and design for manufacturability is important for the particular system being designed. The application of STPA to workplace safety, as well as its use in organizational and managerial analysis, is described in separate chapters of this handbook.

10. Generation of Operational Safety Requirements (including leading indicators of increasing risk and the safety management plan)

The causal scenarios that cannot be eliminated or adequately controlled in the system design must be passed to the system operators so that they can implement operational controls. Leading indicators of increasing risk, described in Chapter 6 of this handbook, can be derived from the STPA analysis generated during the development process or during an operational safety analysis. The goal of these leading indicators is to identify flawed assumptions about the use environment during design, hopefully before any serious loss occurs, and also identify when the behavior of the system changes over time and violates the original design assumptions about usage.

Requirements on system operators and maintainers can be derived directly from the STPA analysis performed during development as both are included as components in the system design. The causal scenarios will provide input to the operational requirements.


SYSTEM DESIGN
Congress and Legislatures
Legislation
Government Reports
Lobbying
Hearings and Open meetings
Government Regulatory Agencies
Industry Associations,
User Associations, Unions,
Insurance Companies, Courts
Regulations
Standards
Certification
Legal Penalties
Case Law
Certification Info
Change Reports
Whistleblowers
Company Management
Testing Reports
Modification/Revision Requests
Design Rationale/Assumptions
Operating Procedures
Test Requirements
Hazard Analyses
Safety Policy
Standards
Resources
Status Reports
Risk assessments
Project Management
Policy
Standards
Procedures
Safety Reports
Work Logs
Inspections/Audits
Problem Reports
Requirements
Safety Constraints
Updates
Hazard Analyses
Review Results
Safety-Related Changes
Procedure Revisions
Software/Hardware Revisions
Progress Reports
Manufacturing
Design & Assurance
SYSTEM TESTING
Congress and Legislatures
Legislation
Government Reports
Lobbying
Hearings and Open meetings
Government Regulatory Agencies
Industry Associations,
User Associations, Unions,
Insurance Companies, Courts
Regulations
Standards
Certification
Legal penalties
Case Law
Accidents and Incidents
Certification Info
Test Results
Whistleblowers
Company Management
Design Rationale/Assumptions
Operating Procedures
Hazard Analyses
Safety Policy
Standards
Resources
Test Reports
Mod Reports
Risk Assessments
Operations Management
Instructions
Modification Reports
Instructions
Configurations
Test Hazards
Problem Reports
Change Requests
Test Operations
Human Controller(s)
Lower Controller
Actuator(s)
Sensor(s)
Physical Process
Maintenance
Problems
Repairs
Revisions
Modifications
SYSTEM FIELDING
Congress and Legislatures
Legislation
Government Reports
Lobbying
Hearings and Open meetings
Government Regulatory Agencies
Industry Associations,
User Associations, Unions,
Insurance Companies, Courts
Regulations
Standards
Certification
Legal penalties
Case Law
Accidents and Incidents
Field Use Metrics
Maintenance Reports
Change Reports
Whistleblowers
Company Management
Modification/Revision Requests
Design Rationale/Assumptions
Operating Procedures
Hazard Analyses
Safety Policy
Standards
Resources
Field Reports
Operations Management
Instructions
Change Requests
Audit Reports
Instructions
Audit Reports
Problem Reports
Change Requests
Field Operations
Human Controller(s)
Lower Controller
Actuator(s)
Sensor(s)
Physical Process
Maintenance
Problems
Audits
Repairs
Revisions

11. Operational safety management including monitoring leading indicators

An operational risk management or safety management system should exist in every operational environment. STPA results can be used to create the operational safety plan and safety requirements. In addition, it can play a major role in identifying leading indicators and monitoring the operation of the system to detect migration to states of higher risk (see the chapter in the handbook on this topic).

STPA can also play a role in incident and accident analysis and change management. Changes such as system design changes (upgrades or mods), context changes (using the system in a different environment or for a different purpose), and operations support changes (logistics supply chain or maintenance practices) need to be considered as well as others.

Planned changes must always be subjected to a hazard analysis. This process should be made easier with the traceability embedded into STPA so that change impact can be traced to specific system hazards or shown not to impact them. Unplanned changes are handled through leading indicators generated from the STPA analysis.

Chapter 4: Workplace Safety using STPA

Nancy Leveson

Workplace or occupational safety has often focused on the worker's behavior and not gone beyond that. A system's approach to workplace safety assumes that human behavior is affected by the system in which it is embedded. The goal is to identify how to design the work environment in order to reduce human error. The application of STPA in this environment is identical to the standard application of STPA on a more technological system, but there are not many examples in the open literature so we have included this chapter of the handbook.

Two examples are shown included: one involves people working with advanced automation and the other involves the less technological workplace hazard involving the use of lockout/tagout (LOTO). In both cases, the major difference between STPA and the standard approach to workplace safety today is that STPA focuses on more than just creating procedures for humans to follow but instead looks at the entire system to identify hazardous scenarios and how to eliminate or reduce them.

Because the first example below describes a real experimental application of STPA in a manufacturing environment, details about the actual assessment procedure used and the cost can be provided.

STPA Example for a Semi-Automated Manufacturing Process

The example here comes directly from the master's thesis of Nathaniel Peper,[17] in which he applied STPA to a real aircraft assembly process where advanced automation (robotics) was being introduced into the workplace. He describes the process as follows:

> "*… the process begins with sub-components for the aircraft entering the factory on the back of a vehicle operated by a worker. Within the factory, two more workers are each responsible for each driving an automated ground vehicle (AGV) by handheld control from an AGV docking and charging location within the factory to connection points on a Product Transportation Vehicle (PTV) for the incoming sub-component. After connecting to the PTV, the two drivers must collaboratively drive the combined Product Transportation System (PTS) around the factory to the location of the sub-component on the first mentioned vehicle. At this point, another group of workers will then work together to move the sub-component from the vehicle to the PTS. The AGV operators will then drive the PTS through the factory to a location at which the sub-component can be prepared for entry into an automated manufacturing cell. Inside the cell are multiple robots and associated support equipment that will perform the automated process of drilling holes in the sub-component and filling the holes with additional components. When the sub- component, the PTS, the robotic cell, and the operators are all ready, the cell operator and controller will assume control of the PTS and move it through the automated process to perform its work statement. During this time, there will also be concurrent work occurring to portions of the sub-component that are not inside the robotic cell perimeter. Once the entire automated process is complete, workers will re-assume control of the PTS and move it down the factory line to the next position for more manual work processes.*"

The accident or loss here is:

[17] Nathaniel Peper, Systems Thinking Applied to Automation and Workplace Safety, MIT Masters Thesis (Leaders for Global Operations Program), June 2017.

A: Death, Injury, or Illness to Humans

General hazards in this workplace:

H1: Exposure to uncontrolled energy (or energy at a level that could lead to a loss)

H2: Potentially injurious movement of the human body (or stress on the body) that could lead to injury

H3: Exposure to toxic materials above a safe level

H4: Exposure to noise levels that could affect hearing

H5: Extended exposure to an environment not providing basic human health requirements

The team refined this set of hazards to one related to the specific application:

H1: Exposure to uncontrolled energy (or energy at a level that could lead to a loss)

H1.1: Violation of minimum separation between AGVs and external objects

H1.2: Violation of minimum separation between PTV and external objects

H1.3: Violation of minimum separation between AGVs and PTV combination and external objects

H1.4: Violation of minimum separation between Robotics and external objects

Other energy sources present in the current design that must be controlled for H1 including electrical, thermal, pneumatic, hydraulic, gravitational, mechanical, and non-ionizing radiation (lasers).

H2 and **H3** can be refined to be:

H2: Potentially injurious movement of the human body (or stress on the body) that could lead to injury

H2.1: Potentially injurious movement of the human body (or stress on the body) that could lead to injury during routine operation

H2.2: Potentially injurious movement of the human body (or stress on the body) that could lead to injury during maintenance, servicing, or troubleshooting

H2.3: Potentially injurious movement of the human body (or stress on the body) that could lead to injury during installation, repair, or overhaul

H3: Exposure to toxic materials above a safe level

H3.1: Workers are exposed to toxic chemicals above a safe level while operating the system

H3.2: Workers are exposed to toxic chemicals above a safe level while servicing the equipment

H3.3: Workers are exposed to toxic materials above a safe level from the manufacturing process

Changing these hazards into safety constraints is trivial and is not included here.

The eight-member cross-functional team that did the actual analysis was composed of members from integration, engineering, operations, and maintenance. All the people involved were considered to be leads or subject matter experts within their scope of responsibility for the system. The facilitator was an MIT master's student who was trained in STPA but had little familiarization with the new manufacturing process being introduced to the plant.

The analysis was completed over the course of a three-week period. A group setting for the assessment was only used twice during the process for less than three hours each, while the rest of the assessment was completed in one-on-one meetings between the facilitator and the respective expert for the portion of the system being discussed.

Because STAMP and STPA are relatively new for the company and entirely new for the team, the first group meeting was used as an instruction period to provide an overview of the methodology being used and then to define the accidents (or losses) the team wanted to avoid, the hazards and system boundaries, and an initial draft of the safety control structure for this application. After developing the initial draft of the safety control structure, follow-on meetings were scheduled with each respective expert to add detail to specific portions of the control structure. Because STPA uses a model of the control structure, the assessment could be focused on specific control loops within the model after the complete model was constructed. This ability to divide up the modeling and analysis process provided for easy scheduling of meeting times and numerous meetings within the available time frames of everyone within the group. During these meetings, details such as control actions, process models, feedback, etc. were added and any conflicts or issues were resolved with the affected parties.

Once the appropriate level of detail was developed for the system safety control structure, the one-on-one or small group discussions continued with STPA Steps 1 and 2. The results of this portion of the analysis were then discussed in another group setting to confirm the individual findings of the group and discuss potential mitigations for the system. This process occurred over the course of three weeks, and the time spent in group, one-on-one meetings, and individual facilitator work totaled approximately 300 hours.

After the hazards were identified, control structures were developed for four different functions and possible configurations of the system, and the RAAs (Responsibilities, Accountability, and Authority) of each controller. The RAAs in the STAMP and STPA context are defined as:

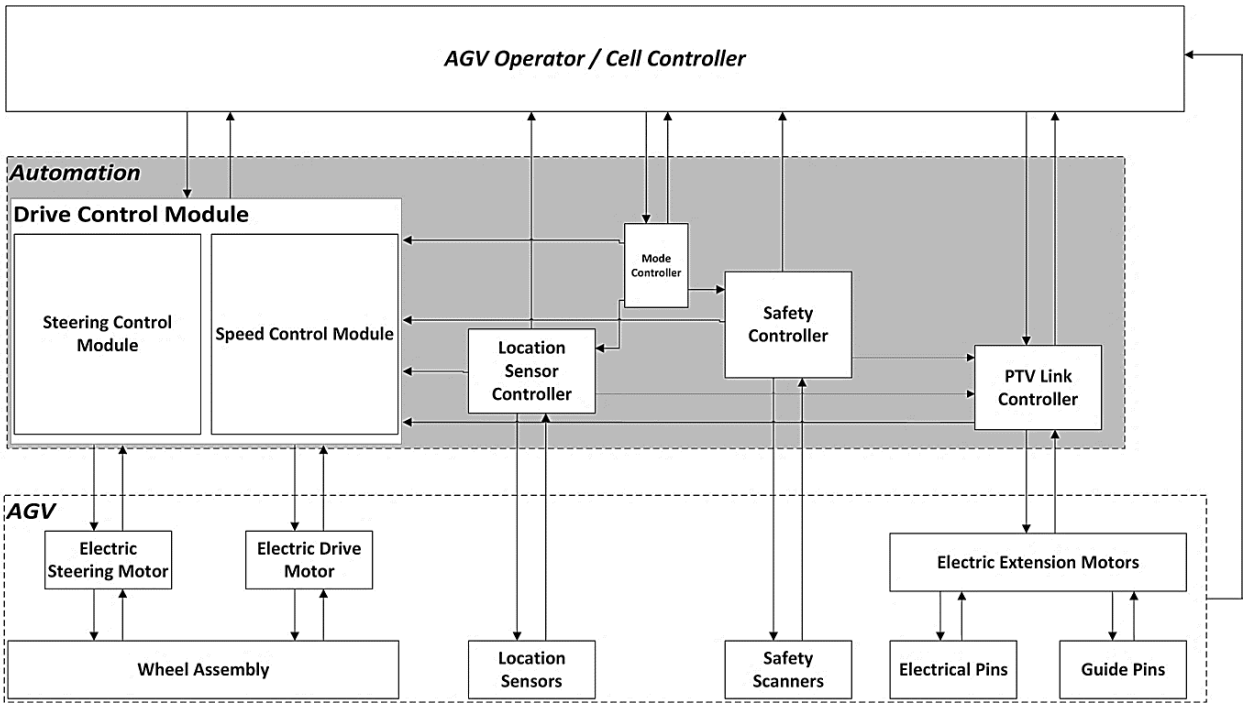
Responsibilities: Basic control responsibilities, process model, and process model variables

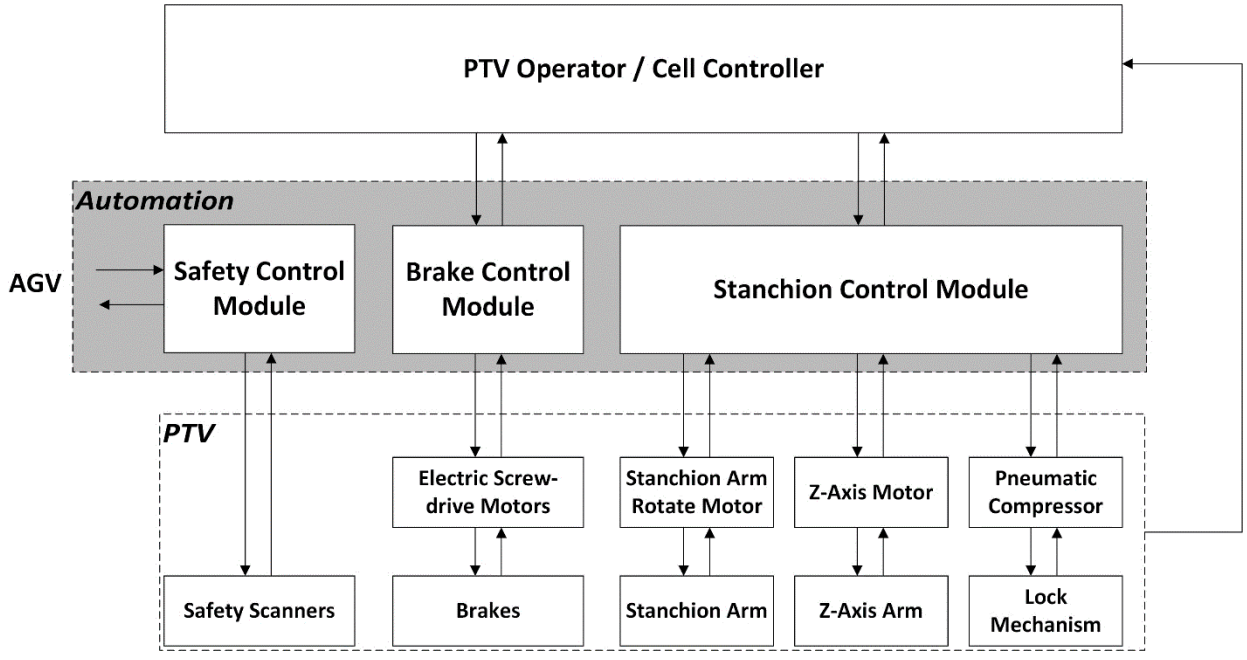
Authority: Possible control actions controller could provide, how they are sent, when they are sent, and where the control action is sent.

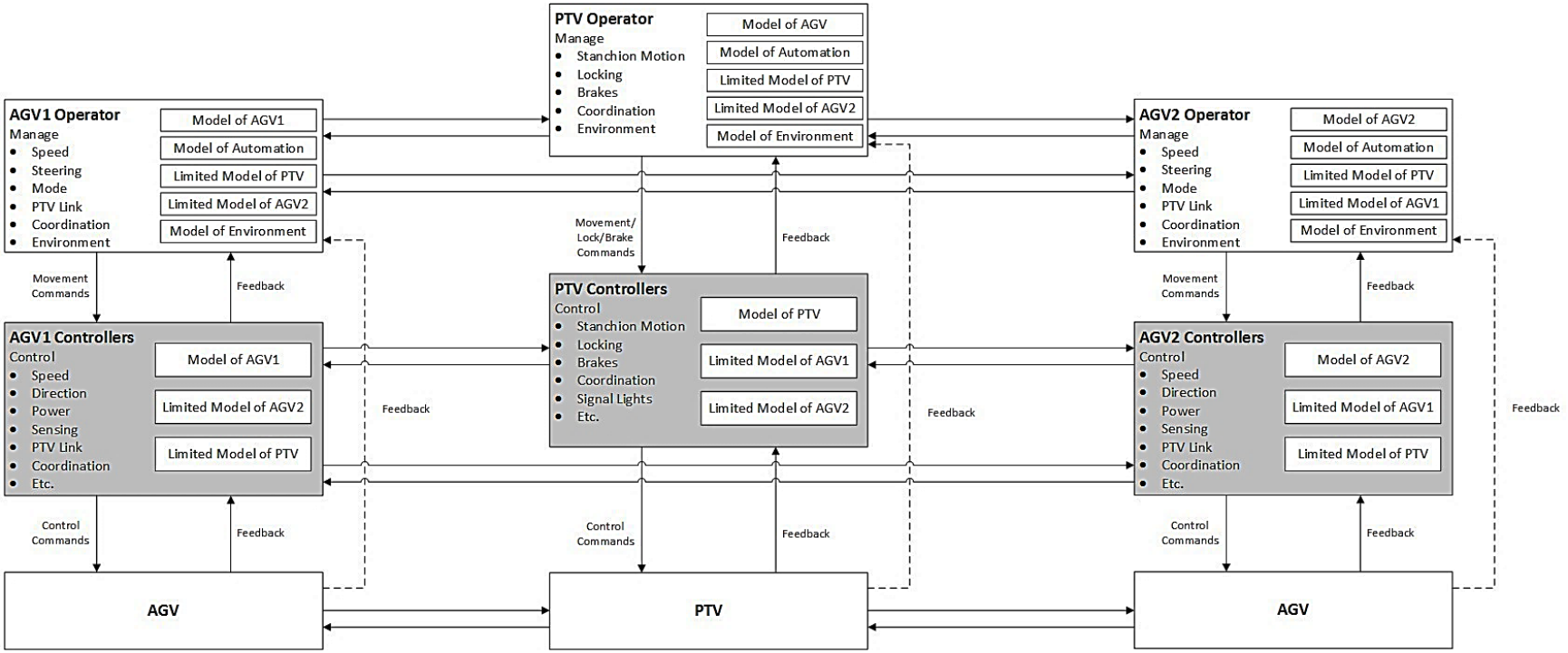
Accountability: What feedback is provided, when it is provided, where the feedback goes in the control structure, how it is given.

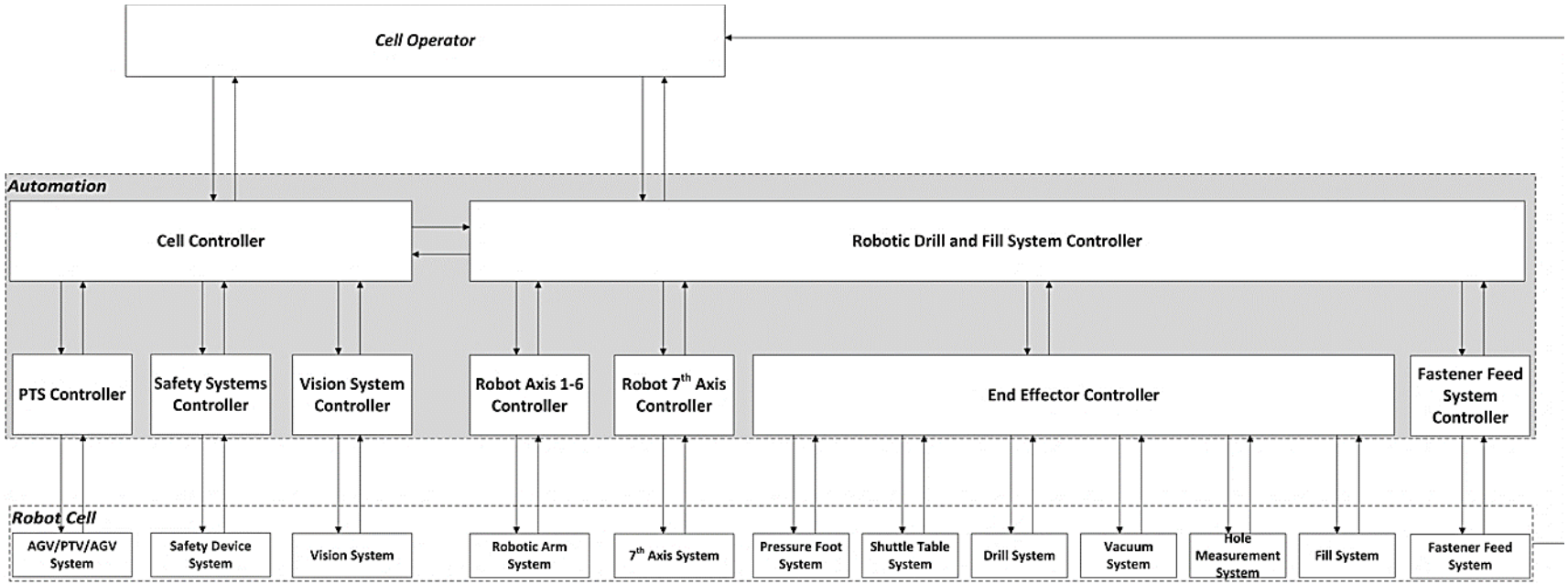
Because this team was new to STPA, they began by developing the control structure as a group to ensure that all of the major details were captured. The models were then refined by individuals and small groups with the appropriate technical expertise on different parts of the system.

The four scenario-based models are shown in Figures 4.1, 4.2, 4.3, and 4.4.

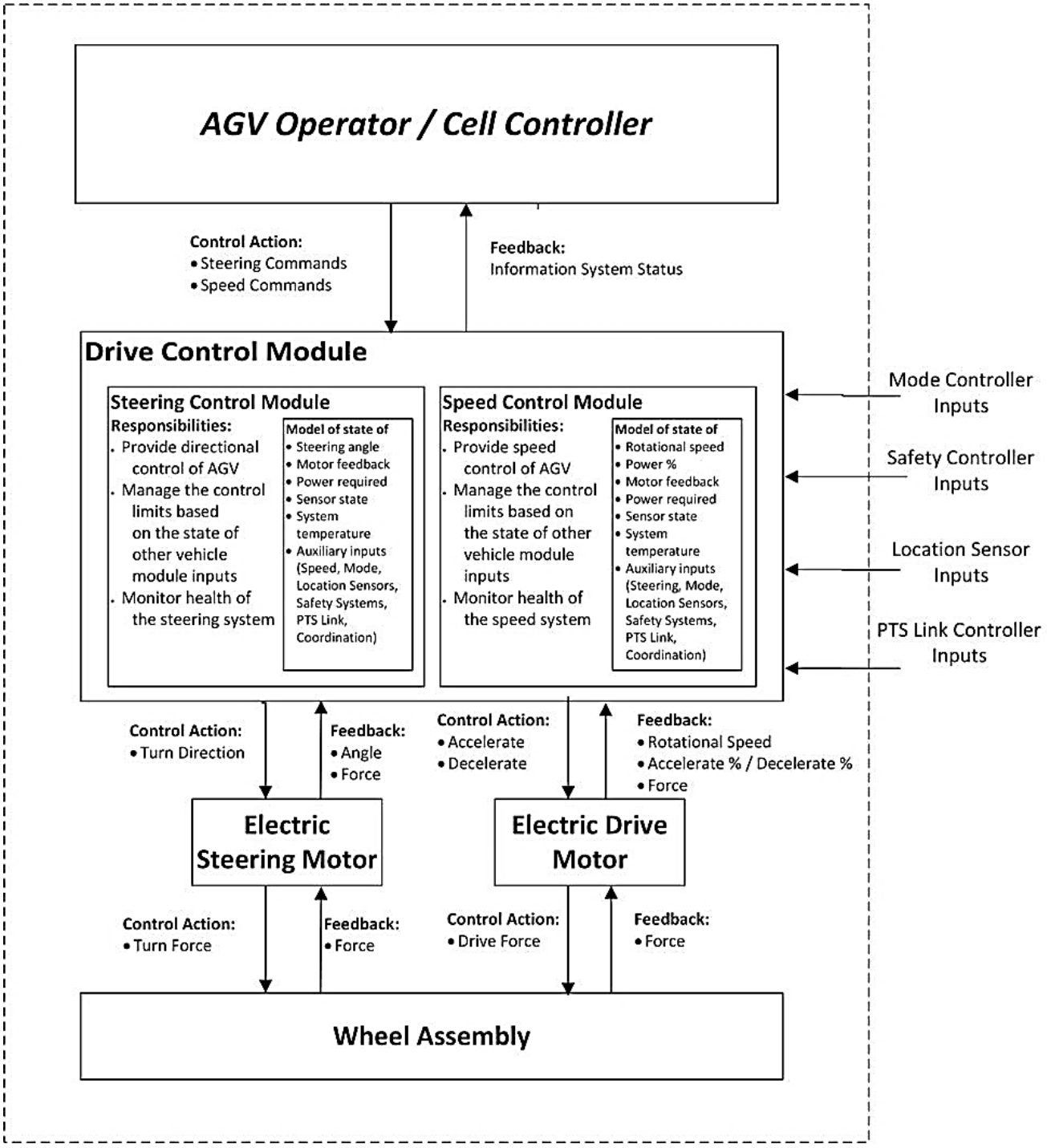

AGV Operator / Cell Controller
Automation
Drive Control Module
Steering Control Module
Speed Control Module
Mode Controller
Safety Controller
Location Sensor Controller
PTV Link Controller
AGV
Electric Steering Motor
Electric Drive Motor
Electric Extension Motors
Wheel Assembly
Location Sensors
Safety Scanners
Electrical Pins
Guide Pins


PTV Operator / Cell Controller
Automation
AGV
Safety Control Module
Brake Control Module
Stanchion Control Module
PTV
Electric Screw-drive Motors
Stanchion Arm Rotate Motor
Z-Axis Motor
Pneumatic Compressor
Safety Scanners
Brakes
Stanchion Arm
Z-Axis Arm
Lock Mechanism


PTV Operator
Manage
Stanchion Motion
Locking
Brakes
Coordination
Environment
Model of AGV
Model of Automation
Limited Model of PTV
Limited Model of AGV2
Model of Environment
AGV1 Operator
Manage
Speed
Steering
Mode
PTV Link
Coordination
Environment
Model of AGV1
Model of Automation
Limited Model of PTV
Limited Model of AGV2
Model of Environment
AGV2 Operator
Manage
Speed
Steering
Mode
PTV Link
Coordination
Environment
Model of AGV2
Model of Automation
Limited Model of PTV
Limited Model of AGV1
Model of Environment
Movement/
Lock/Brake
Commands
Feedback
Movement
Commands
Feedback
Movement
Commands
Feedback
PTV Controllers
Control
Stanchion Motion
Locking
Brakes
Coordination
Signal Lights
Etc.
Model of PTV
Limited Model of AGV1
Limited Model of AGV2
AGV1 Controllers
Control
Speed
Direction
Power
Sensing
PTV Link
Coordination
Etc.
Model of AGV1
Limited Model of AGV2
Limited Model of PTV
AGV2 Controllers
Control
Speed
Direction
Power
Sensing
PTV Link
Coordination
Etc.
Model of AGV2
Limited Model of AGV1
Limited Model of PTV
Feedback
Feedback
Feedback
Control
Commands
Feedback
Control
Commands
Feedback
Control
Commands
Feedback
AGV
PTV
AGV


Cell Operator
Automation
Cell Controller
Robotic Drill and Fill System Controller
PTS Controller
Safety Systems Controller
Vision System Controller
Robot Axis 1-6 Controller
Robot 7th Axis Controller
End Effector Controller
Fastener Feed System Controller
Robot Cell
AGV/PTV/AGV System
Safety Device System
Vision System
Robotic Arm System
7th Axis System
Pressure Foot System
Shuttle Table System
Drill System
Vacuum System
Hole Measurement System
Fill System
Fastener Feed System

Only a sample of the rest of the STPA analysis is shown here. The more detailed segment of the AFV safety control structure used in the example is shown in Figure 4.5.



An example from the UCA table for the AGV follows:

Control Action	Providing Causes Hazard	Not Providing Causes Hazard	Incorrect Timing/Order	Stopped Too Soon/ Applied Too Long
Operator provides drive commands to drive control module	**UCA1:** Drive control module commanded to drive when the movement will violate minimum separation with an object [H1.1]	**UCA3:** Drive control module not commanded to drive when the movement will prevent a violation of minimum separation with an object [H1.1]	**UCA4:** Drive control module commanded to drive before or after a safe path direction [H1.1]	**UCA7:** Drive control module commanded to drive too long when the movement violates minimum separation with an object [H1.1]
	UCA2: Drive control module commanded to drive when a human is handling components that will move [H1]		**UCA5:** Drive control module commanded to drive before a human stops handling components that will move [H1]	
			UCA6: Drive control module commanded to drive after a human starts handling components that will move [H1]	

Causal scenarios for UCA1 (Drive control module commanded to drive when the movement will violate minimum separation with an object [H1]) include the following examples:

Causal Scenario 1: The AGV is driven inappropriately because the operator is not familiar with its operation.

- New operator that is not or inadequately trained

Causal Scenario 2: The AGV is driven toward an external object because the operator does not see the object or misjudges the safe path.

- Operator inattention due to task overload, changes to the environment, or other external factors.
- Operating in cluttered/restrictive areas

- Objects are blocked from view by other workers, the vehicle itself, or the spar load on the AGV/PTV/AGV combination

Causal Scenario 3: The AGV is driven into an external object because the AGV speed/steering settings have changed.

- Modifications to the controller that the operator does not know about or is not familiar with and operating under their previous understanding.
- Degradation to the controller that the operator does not know about or is not familiar with and operating under their previous understanding.
- Modifications to the AGV system that the operator does not know about or is not familiar with and operating under their previous understanding.
-

Causal Scenario 4: The AGV is driven into an external object because the controller has a hardware failure.

- Input is frozen and continues to be communicated even though the operator changes the command.

Causal Scenario 5: The AGV is driven into an external object because the operator holds the drive command for too long due to a delay in the command starting movement in the vehicle.

- Processing delays and computing overloads
- No timeout of control inputs to AGV

Causal Scenario 6: The AGV is driven into an obstacle because the operator is provided incorrect or no information on critical system statuses relied upon for navigation, such as safety scanners and location sensors.

- Operator misses any feedback because it is not easily accessible, small HMI on side of the AGV, next to the ground.
- No status of scanners being on or off
- No status of location sensors being on or off

Causal Scenario 7: The operator drives the AGV into an obstacle that is not detected by the scanners. Possible causes include:

- The object is outside of the safety scanners field of view.
- Obstacles in the factory at the PTV or spar level.
- PTV guide rails above the AGV
- AGV being used for unintended use, such as carrying objects that extend past the scanner FOV.
- The object is in the safety scanners field of view but below the detection threshold
- The object enters the field of view and is impacted by the AGV at a rate faster than the scan and reaction rate of the vehicle
- The scanner capabilities have degraded over time.
- The scanner fails into an unsafe state.

Causal Scenario 8: The operator drives the AGV into an obstacle that is detected by the scanners but does not stop the AGV. Possible causes include:

- Software coding errors do not command the vehicle to stop as intended
- There is a delay in the signal processing due to CPU overload

- The interface with the PTV scanners is inadequate and the PTV scanners cannot send the signal to the AGV Safety Control Modules

Causal Scenario 9: The AGV operator drives the AGV into an external object when the scanners are deliberately turned off.

- The scanners were disabled and the operator did not know it, relying on the scanners to stop the AGV.
- The scanners on the PTV were disabled with no feedback to the AGV operator
- The operator did not see the object.
- Operator is relying on the location sensing module to avoid possible obstacles
- The operator believes the vehicle is in a different mode and expects the vehicle to respond differently
- The AGV has been modified and the operator does not know or understand the updates.
- The commands are delayed and the operator provides the command for too long

In general, the scenarios included such causal factors as hardware failures, incorrect process models (stemming from missing or incorrect feedback), system component interactions, and the role of management and procedures combined with schedule pressures in the factory. After the team completed STPA Steps 1 and 2, they used the results to develop new controls for the system in order to eliminate or prevent entering hazardous states. A few examples follow.

One example comes from the system's heavy reliance on safety scanners to prevent the AGV and product transportation vehicle from colliding with surrounding objects, including workers. If an object is in the path of motion of the system and is not detected by the operator or scanners, there are no other controls designed into the system to prevent the hazard or mitigate the effects of an accident. Due to the massive size, length, and weight of the system and the force of the AGV drive motors, any unsafe control action could result in a severe accident under certain conditions. While operator and workers in the area staying alert to AGV movement around them and AGV safety scanners were the current design control, there are a number of causal scenarios that show them as insufficient.

First, as had already been demonstrated in other parts of the company, workers may bypass safety scanners if they malfunction or are not calibrated or designed properly, preventing the workers from completing their tasks and falling behind on the production schedule. The safety scanners are not integrated in a way that prevents the AGV from operating when the scanners are not in use and there is no notification to the operator or surrounding workers that the AGV is operating with a disabled safety system. While it may not be an issue for the user that turns the safety system off in order to get a job done, if they do not turn the safety system back on the next user will be operating the AGV without knowing that the scanners are disabled.

Another scenario is if an AGV were to encounter an obstacle that the scanners do not protect against. The drive motors for the vehicle do not provide any feedback to the drive controller about actual force being encountered. This would result in the AGV continuing to apply full force into the obstacle until the AGV drive controller receives a different drive command from the operator. An example of this could be something that is outside of the scan volume of the scanners, such as an obstacle above the vehicle but still in the path of the product on top of the transportation system, a scanner adjusted with a resulting gap in coverage around the vehicle, scanner control logic about clearing faults or obstacles when an obstacle is still in the scan volume, multiple controllers managing multiple scan systems with the integrated product transportation system, etc.

AGVs are large and powerful mobile systems that people cannot simply put a fence around to keep workers safe. Given the numerous ways that safety scanners can be and already have been bypassed or do not function as intended, the team discussed multiple new controls to implement. One was a design change to enable the ability for workers to turn the scanners off and provide feedback to everyone in the surrounding area that the scanners have been disabled, such as warning lights or an audible tone. Additionally, the team focused on ways to provide feedback from the AGV to the operator and surrounding workers in general. The original design seemed mostly to assume that the AGV would operate exactly as originally designed and as the operator intended, without helpful and readily displayed feedback to know what the system is actually doing, not just what the operator believes they commanded or what surrounding workers believe the robot should be doing.

Another discussion involved programming motor feedback logic for the drive motors of the AGV as seen in many applications of collaborative robotics. Due to the number of scenarios that resulted in the safety scanners being the only control in place and the number of ways the scanners may not maintain proper control, a secondary method of control for the mobile functions of the system was considered. While this control may still allow minimum separation to be violated as defined in the hazards, the AGV would stop moving and either avoid an accident or lessen the severity of the accident.

While these are just a few examples of the discussion that resulted from the process and results of STPA, the results for the rest of the system were very similar. In order to avoid the defined system-level hazards, there were design changes recommended for communication of process model variables between systems that required coordination and control, for system motor feedback logic for various moving parts, and for increased levels of feedback from the physical system to the human operators and surrounding workers. The team focused on determining how the system could violate the safety controls in place, potentially resulting in an accident, and then how those existing controls needed to change or what new controls were needed to be put in place.

STPA Applied to a Lockout/Tagout (LOTO) Procedure

A second brief example is included to show that STPA is also applicable to more traditional workplace safety hazards. The example is traditional lockout/tagout (LOTO), which is a safety procedure used to ensure that dangerous machines are properly shut off and not able to be started up again prior to completion of maintenance or servicing work. It requires that energy sources be isolated and rendered inoperative before work is started on the equipment involved. The isolated power sources are then locked and a tag is placed on the lock identifying the worker who has placed it. The worker then holds the key for the lock, ensuring that only he or she can start the machine. The goal is to prevent accidental startup of a machine while it is in a hazardous state or while a worker is in direct contact with it.

While this process seems relatively foolproof, in fact, a study conducted by the United Auto Workers found that 20% of the fatalities (83 of 414) that occurred among their members between 1973 and 1995 were attributed to inadequate lockout/tagout procedures.

Accidents:

A1: Person is injured

A2: Product is damaged

A3: Manufacturing equipment is damaged

A4: Delivery is delayed

Hazards:

H1: Person is exposed to hazardous energy [A1]

H2: Product is exposed to excessive or inappropriate hazardous energy [A2]

H3: Manufacturing equipment is exposed to excessive or inappropriate hazardous energy [A3]

H4: Work is delayed at critical junctures [A4]

Safety Constraints:

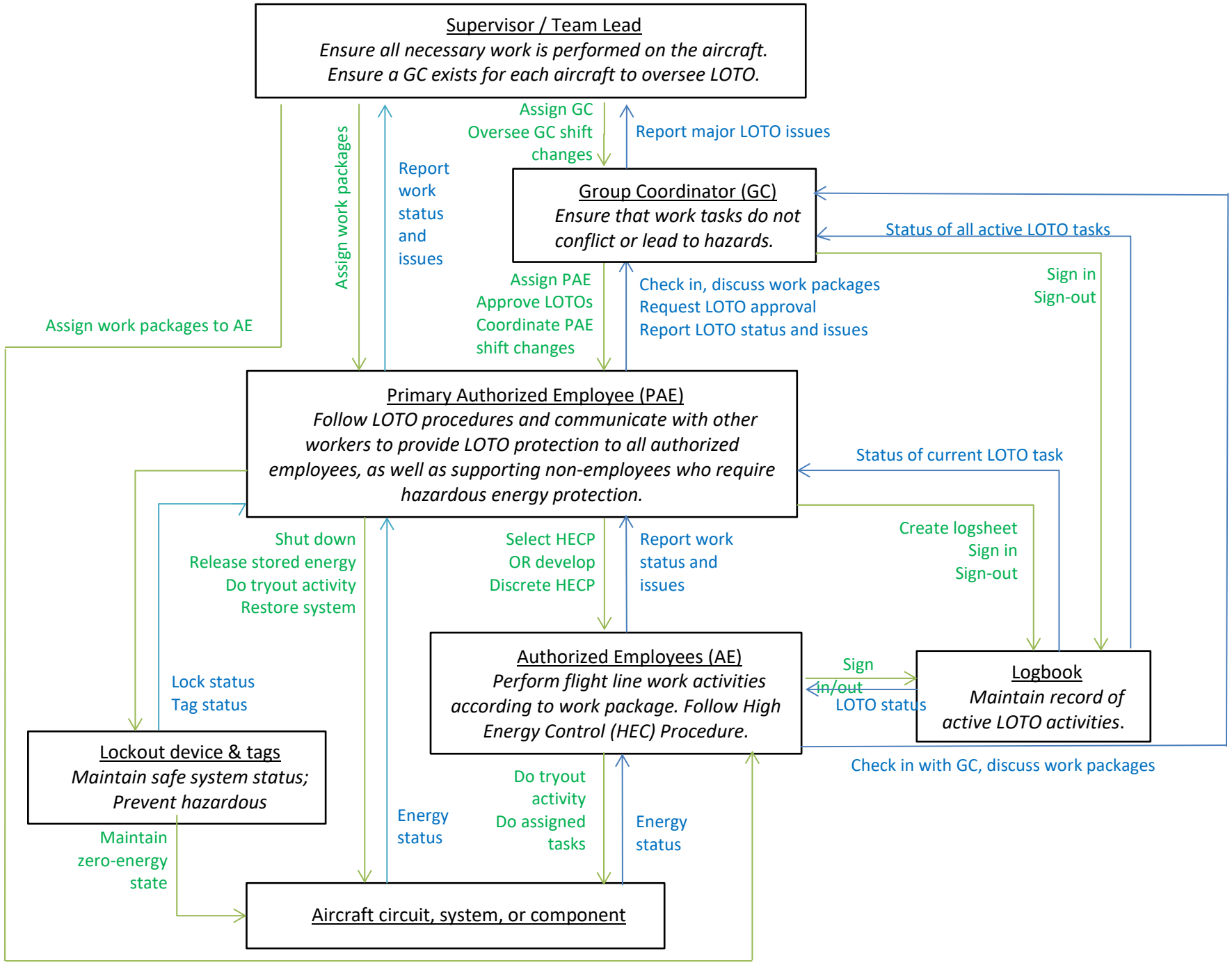
SC1: People must not work in conditions where they can be exposed to excessive or inappropriate forms of hazardous energy. [H1]

SC2: Hazardous energy must only be activated within appropriate and established threshold conditions. [H2]

SC3: Manufacturing equipment must not be used in conditions where it can be exposed to excessive or inappropriate hazardous energy. [H3]

SC4: Work must not be delayed at critical junctures of the process (sign off, lock applied/removed, etc.) [H4]

A generic control structure for this example is shown in Figure 4.6.


Supervisor / Team Lead
Ensure all necessary work is performed on the aircraft.
Ensure a GC exists for each aircraft to oversee LOTO.
Assign GC
Oversee GC shift changes
Report major LOTO issues
Assign work packages
Report work status and issues
Group Coordinator (GC)
Ensure that work tasks do not conflict or lead to hazards.
Status of all active LOTO tasks
Sign in
Sign-out
Assign work packages to AE
Assign PAE
Approve LOTOs
Coordinate PAE shift changes
Check in, discuss work packages
Request LOTO approval
Report LOTO status and issues
Primary Authorized Employee (PAE)
Follow LOTO procedures and communicate with other workers to provide LOTO protection to all authorized employees, as well as supporting non-employees who require hazardous energy protection.
Status of current LOTO task
Shut down
Release stored energy
Do tryout activity
Restore system
Select HECP
OR develop
Discrete HECP
Report work status and issues
Create logsheet
Sign in
Sign-out
Lock status
Tag status
Authorized Employees (AE)
Perform flight line work activities according to work package. Follow High Energy Control (HEC) Procedure.
Sign in/out
LOTO status
Logbook
Maintain record of active LOTO activities.
Check in with GC, discuss work packages
Lockout device & tags
Maintain safe system status; Prevent hazardous
Energy status
Do tryout activity
Do assigned tasks
Energy status
Maintain zero-energy state
Aircraft circuit, system, or component

The table below shows some of the causal scenarios for two UCAs: PAE (Primary Authorized Employee) does not de-energize component being worked on and PAE de-energizes component being worked on after work has started and before it is completed and workers are safely out of the area. Some of these are specified in enough detail to identify fixes. If they are not, then more details will need to be added to the causal scenarios.

PAE does not de-energize component being worked on	…because she believed she was de-energizing the correct component because of insufficient specificity in the HECP. …because she believed she was de-energizing the correct component because of lack of training. …because she believed she was de-energizing the correct component because because of insufficient specificity in the High Energy Control procedures. …because she believed that it was not necessary to de-energize the component because of lack of training. (Training) …because she believed that it was not necessary to de-energize the component because of lack of experience. …because she believed that it was not necessary to de-energize the component because she believed that someone else had de-energized the component …because she believed that it was not necessary to de-energize the component because she believed that she had already done it
PAE de-energizes component being worked after work is started and before it is completed	…because she believed the work had not yet begun because it was not scheduled …because she believed it was not necessary to de-energize the component Because she believed that someone else had de-energized the component. …because she believed that it was not necessary to de-energize the component because she believed that she had already done it (incorrect process model).

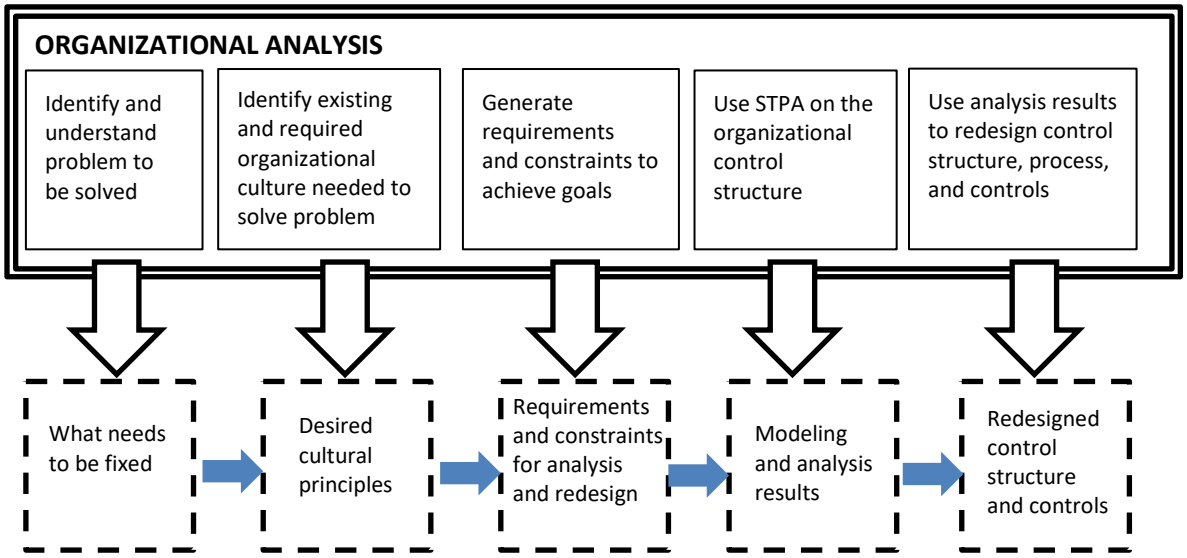
Chapter 5: Organizational and Social Analysis

Nancy Leveson

Applying STPA to an organizational or social system is essentially a system engineering or re-engineering exercise, but the thing being engineered (the product) is the organization itself. In this type of system engineering process, the goal is to determine where the system is now, where it needs to go, and how it can get there. The products and byproducts of the analysis will include:

1. The engineering and business problems that need to be solved, i.e., the goals of the analysis
2. The organizational culture required to achieve the goals
3. Requirements and constraints for the organizational structure
4. STPA analysis of the organizational structure
5. Use of the analysis results to:
    a) Design or improve the organizational structure
    b) Identify risks and leading indicators of risk and
    c) Design (or re-design) the risk management process.

Each of these is described in this chapter of the handbook. The overall process is summarized in the box below.



The application of STPA to managerial and social systems is similar to but differs in some significant ways from its use on engineered products. The most important difference involves the initial understanding of the problem to be solved. While any system engineering or re-engineering process should start with understanding the problem, this understanding process may be more complex when applied to an organization than that used for products. The goals and constraints for an aircraft, automobile, or nuclear power plant are fairly well understood and the first step in analyzing such a system is to get the stakeholders together to select the specific goals and constraints they want applied to their particular product.

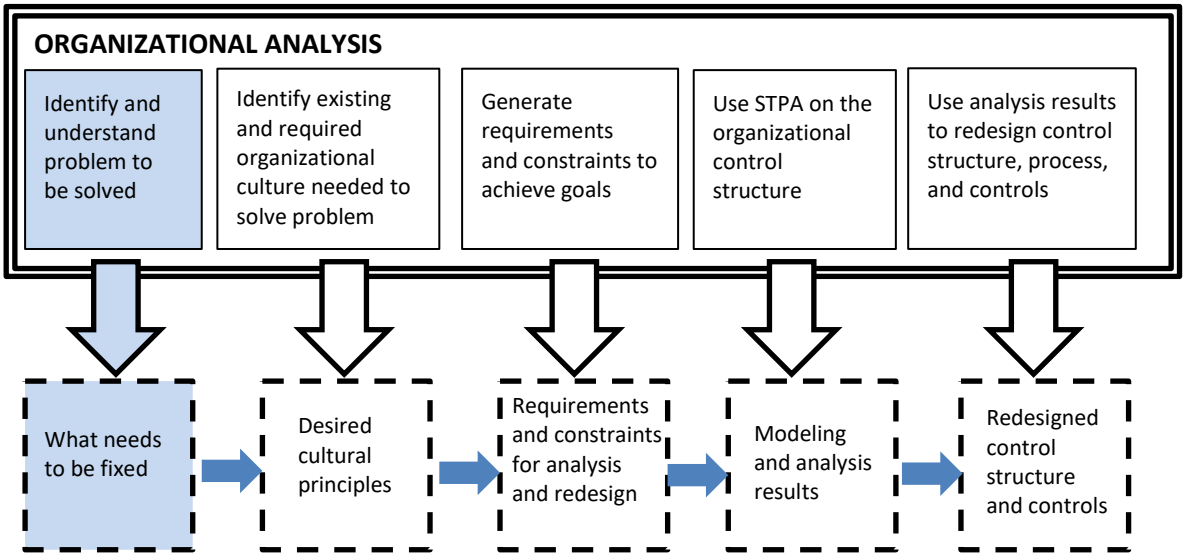
In contrast, when doing an analysis of a social or managerial system (which usually already exists), the first step is to identify and understand why the goals are not being achieved currently or how they

could be better achieved. Because most of the examples in this handbook focus on safety, the example here will examine a different property in order to illustrate how to use STPA for an emergent property different than safety or security.

The example used here is system engineering effectiveness, namely, getting products out the door on time, within budget, and satisfying the needs of customers. An assumption is made that the company being considered produces complex, engineered products for external customers, including both government and commercial entities.

Because the proprietary information obtained when we have performed this process on real companies cannot be divulged, generic results of the process and of its use on real companies is presented here instead.

Identifying the Engineering and Business Problems that Need to be Solved



Although not always followed, the first step in a successful solution of a problem is to understand the problem that is to be solved. Identifying the solution before spending time understanding the problem to be solved is not very likely to lead to success.

Interviews with stakeholders and key personnel are a useful way to identify the problems needing to be solved and the perceived causes of those problems as is simple observation. The resulting information is then used for an analysis of the current system and potential improvements. Information about how to do such interviews is contained in the social science literature, but we have been able to get the information we need using a simple and straightforward interview process that first describes our goals, promises anonymity, and then probes with simple questions, allowing lots of freedom for the interviewee to steer the answers in the direction they want.

For the purpose of this example, let's suppose that the interview process asks about the reasons why system engineering may not always be successful in their company. The interviewees should at least include management, engineers, and, if possible, customers. Given a focus on system engineering, the results might turn up the following perceived explanations of the problems and why they are occurring:

- The projects do not always have the expertise and skills at the time they are needed. Skill development is not as strong as it needs to be.

- The proposal writing and contract negotiations result in unrealistic budgets and/or schedules. So the projects start out behind, which leads to cutting corners in product development and, in the end, simply makes things worse.
- Just as determining the problem to be solved in the re-engineering process we are describing here is critical, so is determining what problem is being solved before creating a solution is important in satisfying customer needs and in avoiding having to backup and redo work later in the system engineering process itself. But in this example company, too little effort is usually spent in upfront planning and the early stages of development. This includes too little time understanding customer needs and the environment in which the product will be used. Engineers waste time on resolving problems that arise from bad plans and from not identifying the critical assumptions and requirements about the desired product in the early stages of development. Hasty decisions are made that cost time and money later and often result in a product that is not as useful as originally intended.
- Lack of early planning leads to being late and to continual re-planning, which wastes time and creates project "thrashing," where so much time is spent in meetings and re-planning that forward progress is seriously degraded and also to budget and schedule problems. The haste in which changes start to be made simply accelerates the problems as change impact analysis and change management starts to degrade.
- Project risk analysis and mitigation is perfunctory and becomes a "check the box" activity. Few resources are assigned to identify and mitigate risk. Often, risk analysis is given a lower priority than other activities and delegated to new engineers who do not have enough experience to do it well. In addition mitigation of risks is often unfunded and the steps to achieve it are not in the master schedule.
- System engineering gates and milestones were created to ensure readiness for the later steps in the process. While most everyone understands the need for the gates, pressure builds to remain on schedule. Longer term schedule and quality concerns are subsumed by short-term focus on predefined schedules. Project managers will often skip over milestones ("gates"), assuming they will be covered in the next phase. The projects never catch up and end up fixing problems late, which is, of course, always enormously expensive. Budgets and schedules are blown to pieces when changes are needed late in the project. Even worse, many, if not most, of the critical decisions are made in the early stages of development and cannot be undone without enormous and usually impractical redesign. At that point, the only viable solution usually is to create workarounds, which result in a deficient product. Decisions and changes made in haste often create even worse problems than those being solved.
- Related to the problems involved in rework is the often inadequate recording of design intent and rationale. Lack of such documentation can lead to solutions that were earlier discarded as inadequate being suggested and tried again.
- All of this leads to heroism and cowboy engineering becoming the standard and being highly rewarded by management. There is accountability for satisfying schedule and cost goals, but there is little accountability or high rewards for first time quality, by which is meant few changes required because of deficiencies in earlier stages of development.
- Inadequate learning may result from both success and failure along with inadequate feedback to provide continual improvement. There is too much emphasis on following rigid processes as the solution to all problems. Managers think that if they just institute the right process, all the system engineering problems will be solved while forgetting that people are involved. Management needs to be more intelligent in the way they use process as the solution to a problem, and, alternatively,

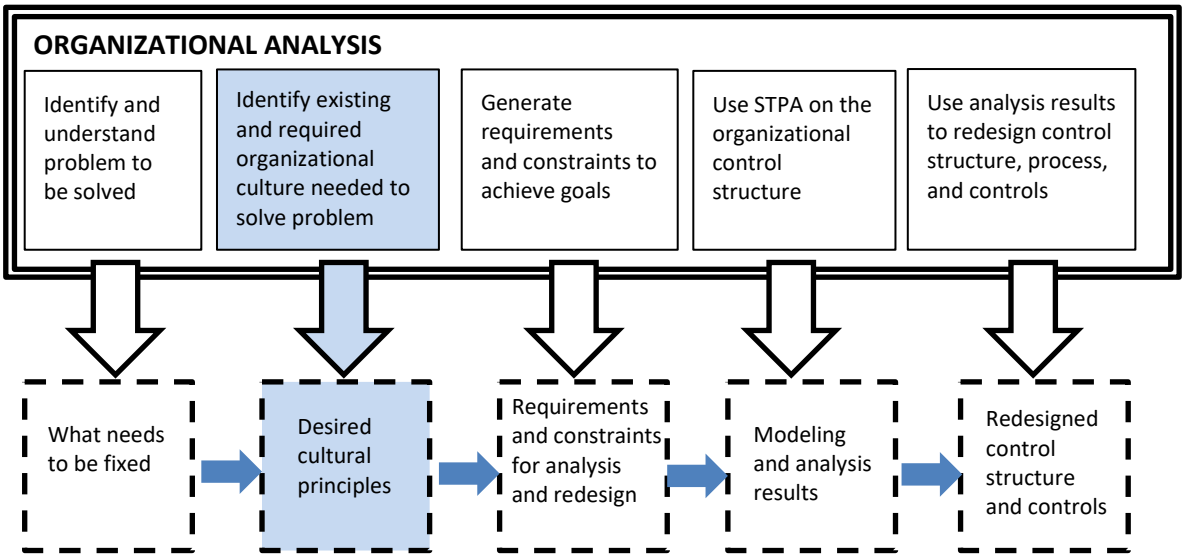
how they develop and support the human components of the development process. The company is good at lessons "observed" (collecting data) but not lessons "learned."

- Learning requires innovation, risk taking, and flexibility (in contrast to rigorously enforcing a specific process). However, there is not enough empowerment and support for those factors that promote innovation in the workforce. A common view expressed in the interviews is that employees are afraid to take risks and innovate because there is little reward for this and, in fact, often punishment ensues. So they don't step outside of what they have always done and been told to do. The fear of failure and resulting punishment can stifle innovation. The current engineering structure is described as not being able to handle disruptive innovation and as having little ability to handle change.
- Another reason for lack of innovation that often comes up in our interviews is that the bulk of engineering innovation and energy is spent resolving program problems arising from poor practices or deficient risk management and not on efforts to create better products.
- There is inadequate design focus on producibility and supportability (maintainability), resulting in cost and quality factors in the inevitable product evolution.
- Too much emphasis is placed on ensuring safety and security in a completed design or architecture and not enough on building safety and security into the product from the beginning.
- For companies that subcontract parts of their product, we often hear about ineffective supplier oversight leading to schedule slips and the need for rework.
- There is lack of independence of technical decisions from programmatic and business strategy considerations when schedule is in conflict with quality. This may be coupled with a lack of open, honest communication and independent paths to elevate technical risk and issues.
- Engineers and managers are not getting the information and feedback they need for good decision making. Sometimes they are ignoring the information they do have. Feedback loops are either nonexistent or ineffective. There are complaints that feedback is often received but ignored. There is a well-known phenomenon called *confirmation bias,* which means that people tend to believe information that supports their hypotheses and to ignore information that does not. Confirmation bias is related to *defensive avoidance*, which occurs when information one does not want to hear is disbelieved or ignored.
- Some problems often expressed, especially by those in manufacturing or by customers, are related to the relationship between the designers and the product assemblers or between the designers and the operators of the systems they design. Common problems are associated with deficient work instructions and lack of feedback from mechanics to design engineers.

While the number of problems described above may seem surprising, in fact, they are fairly common in industrial system engineering organizations.

During the interviews, we usually also try to learn about the current organizational culture.

The Organizational Culture Required to Achieve the Goals



To discuss organizational culture, we need to start with a working definition. Edgar Shein's definition is used here:

> *Organizational Culture is the shared values and deep cultural assumptions of a group or organization that provide the basis for decision making.*

To most effectively make important changes in an organization, it is necessary to change the culture to one that promotes those changes.

During the interviews, we define what we mean by organizational culture and ask the interviewees what cultural changes they think are needed in their organization. Their answers reflect the desired behavior they think the culture should promote. Some examples of the feedback we might get in interviews related to system engineering effectiveness and the organizational culture needed to improve it include:

- Less focus on schedule as the primary driver of decision making
- Ability to say "no" to customers and within the company. Marketing and sales answers are provided instead of engineering answers.
- Thinking beyond the immediate program and customer and across the programs
- More effective, clear communication
- Learning more from both failures and successes
- Spending more time up front in order to save time later
- More collaboration
- Valuing engineers and treating them with respect
- Believing that the work is not done until the customers can use the product and are happy
- Not being afraid to ask for help
- Openness and honesty about the product; don't shoot the messenger

- Thinking about the bigger company goals and not just locally
- Not “postponing the pain” and letting the short-term take priority over the long-term.
- Ensuring that the risk mitigation plans are properly resourced and not just communicated.

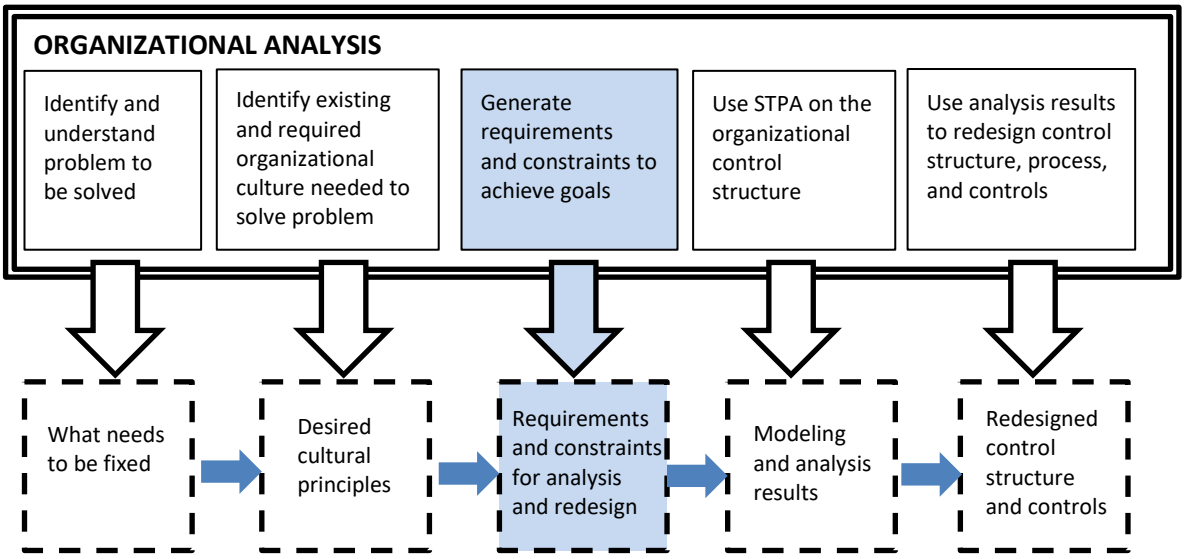
Clearly these are related to the problems identified in the interviews. The next step in our usual process is to take these behaviors, along with the concerns elicited in the interviews and listed in the previous section, and restate them as cultural principles (the shared organizational value system) that would provide the type of organization desired. The results are stated as a set of prevailing beliefs or values that, if implemented in the organizational design, would go a long way toward solving the identified problems. Here is an example set of cultural principles that might be created from the example interview results presented so far:

- Quality and Safety: Increasing quality and safety leads to decreasing cost and schedule and, in the long term, to increased profits. Safety is a key determinant in productivity and profitability.
- Planning: Doing good system engineering upfront saves costly rework later on. Good engineering should not require excessive effort and rework.
- Workforce and Skill Development: Effective workforce and skill development and utilization increases morale, decreases cost, and helps develop our talent base for the future.
- Technical Independence: Technical decisions will be better if they are made independent of program execution decisions. Thinking beyond the immediate program will lead to long-term benefits for the enterprise as a whole.
- Communication: Effective, clear, consistent, timely communication and the sharing of information are essential to achieving engineering goals.
- Learning and Improvement: Effective engineering requires continual learning and improvement, including learning from both success and failure. Building knowledge is as important as building products.
- Innovation: Investment in new knowledge and technology (innovation) drives growth and productivity.
- Collaboration: Successful engineering requires working across domain and structural boundaries, with all the applicable engineering disciplines included.
- Risk Management: Risk assessment and management are key to achieving performance goals.

Generating a list of desired cultural principles (values and beliefs) is only the starting point; it will require review, editing, and (when completed) buy-in from the leadership team and the engineers. The goal should be to start a discussion and begin a search for ways to change the company culture.

Once a consensus has been formed on the engineering culture that will help to achieve the goals, then the path forward in terms of requirements to implement that culture becomes more feasible to plan and implement.

Generating the Requirements and Constraints to be Implemented in an Effective Organizational Structure



A critical step, after identifying the goals for any system engineering effort, is to create requirements. These are generated from the results of the problem identification process and the elicitation of the desired engineering culture. Here is a set of example requirements that might be generated using the sample analysis results so far in this section of the handbook. They are grouped under a restatement of each of the agreed-upon cultural goals.

Quality and Safety: Increasing quality and safety leads to decreasing cost and schedule. Safety is a key determinant in productivity and profitability.

- Each project shall have people with the appropriate skills and expertise needed to successfully achieve the project goals.
- Safety and security analysis shall start at the concept analysis stage and continue throughout the development process using safety and security analysis to inform design decisions as they are made.
- Gate (milestone) controls shall include a quality and safety analysis before they can be considered satisfied.
- Project managers shall be rewarded for first time quality[18].
- All product development shall emphasize producibility and supportability.
- Validation shall be emphasized at each stage of development. Attempts to bypass or decrease validation steps must be prevented.
- The company shall oversee the quality and safety of subcontractor products.

[18] The importance of incentivizing the "right" behavior is discussed in the chapter on Safety Management Systems. For example, giving managers bonuses for meeting timelines can lead to decreases in quality.

Planning: Doing good system engineering upfront saves costly rework later on. Good engineering should not require excessive effort and rework.

- Predicting future performance in proposals shall be, at least in part, based on executability and engineering feasibility rather than simply external expectations such as customer goals.
- Detailed project planning and system analysis shall be done at the beginning of the project including detailed risk management and contingency planning; thorough understanding of customer needs, the problem to be solved by the product, and the environment in which the product will be used; and early activities to understand and mitigate risks such as simulations, prototyping, etc.
- When changes are required, rigorous change management shall be implemented.
- Subcontracting decisions shall be made with appropriate engineering input.
- Engineering plans shall include schedule and cost margins and/or avenues for flexible reallocation of resources to allow for handling inevitable 'technical unknowns."
- Plans shall be adaptable in the face of changing customer needs.

Workforce and Skill Development: Effective workforce and skill development and utilization increases morale, decreases cost, and helps develop our talent base for the future.

- Forecasting of skill needs shall be done, leading to providing appropriate training, mentoring, and growth experiences. System engineering in particular shall be trained and made a core piece of each project.
- When possible, leaders shall be selected on the basis of domain-specific experience. Such experience shall be developed for future leaders.
- Management shall take measures to ensure the workforce is appropriately rewarded and their contributions valued.

Technical Independence: Technical decisions will be better if they are made independent of program execution decisions.

- An independent authority (not the project manager) shall be responsible for determining whether a gate has been passed.
- Technical decisions shall be independent of programmatic and business strategy considerations when schedule is in conflict with quality. An independent technical authority shall be responsible for ensuring this independence.
- A higher authority than the project manager shall be responsible for resolving conflicts between the project manager and the independent technical authority.
- There shall be independent paths for elevating technical risk and other engineering issues.

Communication: Effective, clear, consistent, timely communication and the sharing of information are essential to achieving engineering goals.

- Better communication between Engineering and Business Acquisition shall be instituted, including feedback on the executability of proposals that are in preparation and the success or serious problems on contracts that have been completed (or canceled).

- Feedback and information channels shall be created to ensure that decision makers have the information they need to make appropriate decisions.

Learning and Improvement: Effective engineering requires continual learning and improvement, including learning from both success and failure. Building knowledge is as important as building products.

- *Post mortems* shall be produced by project managers in order to learn from both success and failure. Project managers shall be rewarded for honesty and thoroughness in these *post mortems*.[19]
- A project management information system shall be established that includes not just data from past projects but lessons learned from both successes and failures.

Innovation: Investment in new knowledge and technology (innovation) drives growth and productivity.

- Innovation, flexibility, and appropriate risk taking shall be encouraged and rewarded.
- The organization shall provide resources to produce innovative solutions to new technological challenges and opportunities and the inevitable technical issues that arise on most programs.
- Cultural barriers that prevent engineers from taking innovative risks shall be identified and removed.

Collaboration: Successful engineering requires working across domain and structural boundaries, with all the applicable engineering disciplines included.

- Working across domain and structural boundaries to promote collaboration shall be promoted and optimized.
- Work groups shall be established across domains and structural boundaries with all applicable domains included.

Risk Management: Risk assessment and management are key to achieving performance goals.

- Risk identification and assessments shall be identified by the most knowledgeable subject matter experts, clearly communicated to all of the affected personnel, and lead to a well-defined set of planned and resourced mitigations that are enacted in line with the program objectives and customer needs.
- Risk mitigation shall be planned from the start of the project based on the early risk assessment and any updates as experience is gained. Risk mitigation measures shall be funded and incorporated into the master schedule.
- Risks shall be clearly communicated to all affected personnel.

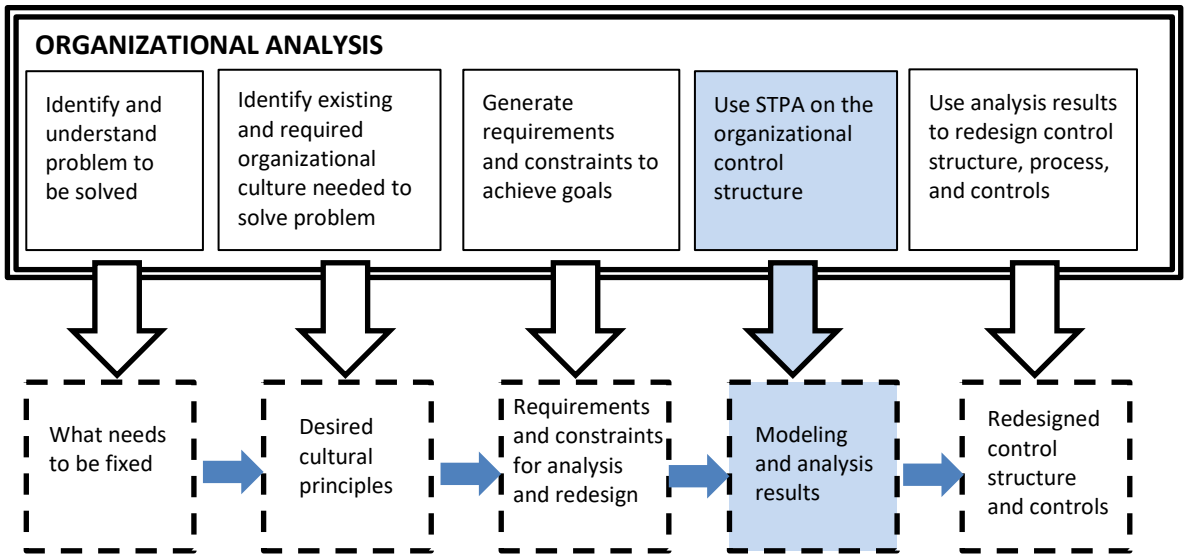
[19] Brutally honest project *post mortems* have been cited as one of the reasons for Microsoft's success. Time is provided for detailed analysis of the project results and leaders are expected to be honest about the reasons for any successes and failures. The military has a similar concept called After Action Reviews. Learning from one's own experience (and that of others) is a requirement for future success. Not rewarding people for honestly evaluating their own performance and sharing lessons learned may lead to new people being promoted that may make the same mistakes and thus inhibiting learning from experience.

- Risk management plans shall include cost and schedule margins sufficient to allow for “unknown unknowns.”
- Risk management shall have appropriate controls such that realistic risks cannot be ignored by managers.
- Risk management shall be properly resourced.

These are simply examples of requirements that might be generated from the agreed upon cultural principles to act as the foundation of the organization. Others are possible.

Once the requirements have been agreed upon, the actual modeling and analysis can begin.

STPA analysis of the organizational structure



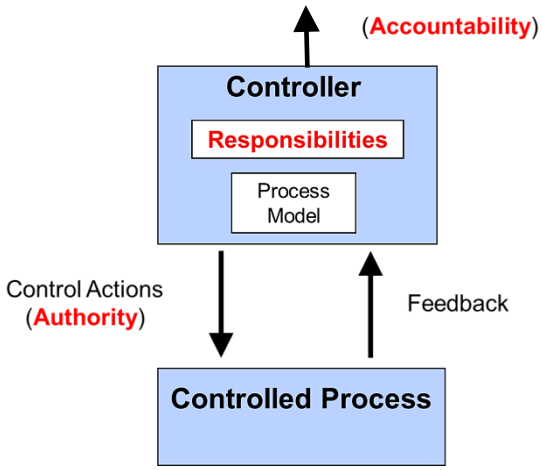
The next step in the process is to model the current system engineering control structure, to identify where the identified requirements are currently implemented (if at all) or at least where responsibility is assigned for these activities, and to determine why the concerns identified (in phase 1) are not already being adequately addressed and the requirements implemented. The results can be used to suggest changes that will better implement the requirements in the organizational design.

When the modeling is complete or at least at the point where the general control structure is understood, then the hazard analysis on the structure can begin. The hazard analysis includes the generation of leading indicators to provide early identification of assumptions about system engineering in the organization that are not true or that may no longer be true as the organization and environmental factors change over time. These leading indicators should also provide a way to identify how well the concerns and goals are being addressed.

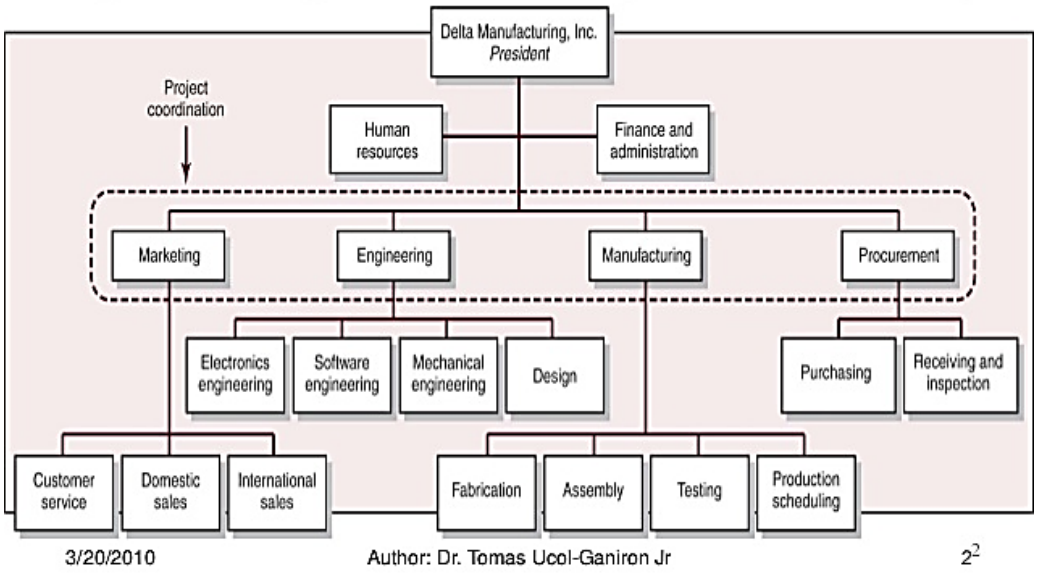
We have found during the modeling and analysis process that people often know the solutions to their problems, but are so overwhelmed with day-to-day schedules and problems that they don’t have the time to implement them. We have also found that use of the hierarchical control structure model is invaluable in helping with communication within the solution-generating groups and also with generation and evaluation of potential redesign solutions. Having a common model to guide discussion

and problem solving is by itself very helpful in finding solutions to problems. Sometimes, for example, without any sophisticated analysis, it can be observed that some entity is trying to control another without any feedback to guide their control (management) actions. Adding effective feedback may go a long way toward solving the current problems.

The following shows the generic loop structure shown previously in this handbook but augmented with the standard management requirements of responsibility, authority, and accountability.



Below is shown an example system engineering structure that might result from the modeling effort.



Notice that project coordination is simply assumed to occur in among four same-level controllers in this example without responsibility being assigned to any individual. It is unlikely to be an appropriate responsibility of the president of the company.

The model, of course, would include specification of the responsibilities, authority, and accountability for each function in the structure. For example, assigned responsibilities for general controllers in a product development company might include:

Human resources:

- Training, hiring, workforce and skill development
- Project management
  - Project planning and concept development
  - Technical decision making
  - Integration of engineering efforts, communication within projects
  - Risk management and contingency management
  - Milestone and schedule management
  - Budget/resource management
  - Managing subcontracts
  - Team dynamics/communication
- System engineering
  - Implementing technical decisions
  - Specialty engineers
  - Implementing technical decisions, project requirements in component design
- Business development
  - Bringing in new business
  - Bidding on projects
- Program management
  - Program/Project risk management
  - Overseeing project progress and success, workforce management
  - Managing subcontractor relationships, supply chain
- Executive Management
  - Implementing technology/process improvements
  - Business Risk management
  - Forecasting skill and engineering needs in the future
  - Ensuring current and future skill and workforce needs are satisfied
  - Workforce “management”
- Research and Engineering Improvement
  - Innovation, new technology/process improvements
  - Identifying future business and technology trends, skill needs
- Quality assurance and system safety
  - Oversee product quality
  - Evaluate product safety

Appropriate next steps are (1) informal review sessions to ensure the correctness of the control structure as well as to identify weaknesses, (2) analysis of the control structure model using traceability from requirements to the control structure responsibilities, and (3) application of formal STPA analysis where useful. The goal is to identify where and how each of the requirements (specified previously) are being realized in the organization control structure and any weaknesses in the control structure with respect to these requirements. Examples are provided below.

The first step is to validate the model through review by experts on the current organizational design. Multiple reviews may be needed as reviewers may only be familiar with part of the structure.

Once the overall control structure is validated, a next step is to examine the control structure with respect to the specific problems identified in the interviews as well as to analyze the control structure design to identify general changes that may be needed to improve decision making. Often there are things that stand out immediately, such as the complexity of the structure; multiple people controlling the same process or having the same overlapping responsibility; lack of appropriate or independent oversight of decision making; or missing feedback paths (i.e., people may be making decisions without all the information needed or with inaccurate information).

To find further weaknesses, tracing from the requirements to the control structure can be done. The table below shows a few examples of potential tracing results for the requirements included above.

Requirement	Control Structure Component(s)	Weakness Identified
Risk Management	Project Management, Program Management	• Cost and schedule margins are often not sufficient to deal with contingencies • Risk identification sometimes is incomplete and important risks ignored
Technical Independence	Project Management, Program Management	• Technical decisions are not independent of programmatic and business strategy nor are they independent from the project manager
Quality and Safety	Quality assurance, project management, program management, executive management	• Safety is assurance-oriented and is not performed by or in close collaboration with the development team. • Much of quality assurance is “checklist” and perfunctory. • Gate (milestone) controls do not include a quality and safety analysis • Project managers are more highly rewarded for schedule and budget performance than for quality and safety. • Producibility and supportability are not emphasized • Contractors are minimally supervised with respect to quality and safety

The weaknesses identified will provide guidance for the redesign activities to improve the satisfaction of the requirements. Some improvements will require few structural changes. For example,

clearly the inadequate supervision of contractors with respect to quality and safety requires only a simple assignment of responsibilities to the manager of the contractors and probably assistance from quality assurance and safety. Other fixes may require more extensive structural changes. If policy for how such contractor supervision should be done is inadequate or missing, for example, then some effort on policy creation and changes will be needed.

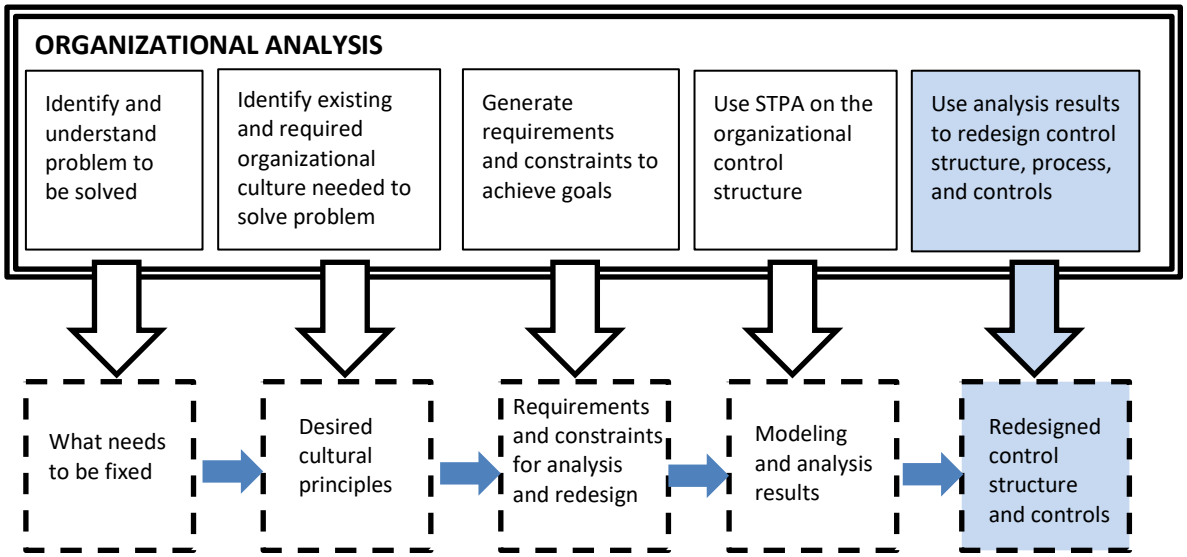
As an example of a more complex change, technical independence will require new management approaches, such as setting up an independent technical authority[20] as well as new communication links and a decision-making process for resolving conflicts. Independence tends to fade over time, so leading indicators would also be appropriate here (see the next section on leading indicators).

A final example is that separating safety from engineering decision-making tends to make it an after-the-fact assurance or compliance effort with almost no impact on the design effort. Finding safety or quality problems late is very expensive to fix and separating safety responsibilities from basic engineering is a mistake, although very common. Determining the best strategy to optimize the interaction could require extensive changes in responsibilities, communication, and structure. For example, there might be a safety engineering component in project engineering and a separate one in quality assurance, each having very different responsibilities and performing different activities.

In some cases, it may be appropriate and useful to apply a full STPA causal analysis on the control structure or to parts of it. As an example, it may not be clear exactly why risk management has not been adequate as most companies put effort into performing this important task. It might be useful to identify all the potential causes, such as lack of information about what the risks actually are (flawed mental models), a culture that emphasizes positivity and not potential risks, inadequate training in risk assessment, poor procedures, pressure from management to downgrade risks on important projects, etc. After generating all the potential causes, a study might be initiated to determine the operative causes in this organization and what changes might be useful to eliminate or control them. While some hypotheses could be formulated in an *ad hoc* manner, using the step-by-step process in STPA will improve completeness and reduce the possibility that important causal factors are missed. There may be different reasons for potential deficiencies in each type of risk management control action in the organization. For example, not providing adequate emphasis on identifying some types of risks may involve different causal factors than not providing adequate contingency plans. We have found that using STPA to identify risks is much more effective than the usual brainstorming process by experienced employees.

[20] The Independent Technical Authority in the U.S. Navy SUBSAFE program has been extremely effective in eliminating accidents covered by the program for the past 50+ years. You can read more about it in Nancy Leveson, *Engineering a Safer World*, 2012, chapter 14 and in the next chapter on Safety Management Systems.

Using the Results in an Organizational Redesign Process



A final step in the process is to use the analysis results in a redesign and risk management process for the control structure as a whole. Part of the risk management in the design of controls involves a leading indicator program. Applying STPA to the control structure or parts of it can be helpful in identifying leading indicators of increasing risk, which is discussed in the next chapter.

Chapter 6: Identifying Leading Indicators Using STPA[21]

Nancy Leveson

Risk management is a critical part of any program and the programmatic and organizational risks need to be carefully identified and managed. Risk assessment is usually performed in an ad hoc fashion, with a group of experts brainstorming what appear to be the risks in a program and then assigning risk levels to them, often rather arbitrarily. STPA provides a structured process on which to base a risk management program.

When leading indicators are used for risk management, they are usually also created in an ad hoc manner. Using a more formal, structured system-thinking process may be more effective, just as it is more effective when considering product safety and other emergent properties as described earlier in this handbook. Using STPA to create leading indicators also has the advantage that the resulting indicators are directly traceable to hazards and accidents. This chapter describes such a structured approach to identifying leading indicators that includes using the results from STPA and other results of the overall system engineering process. The topics covered include:

- What is a leading indicator?
- What do people use now for leading indicators?
- What are assumption-based leading indicators?
- How to
  - ✓ Identify appropriate and effective leading indicators using STAMP and STPA
  - ✓ Generate assumptions on which to base leading indicators
  - ✓ Use the assumptions to create an assumption-based leading indicator program
  - ✓ Integrate leading indicators into your risk management program
- Feasibility and final thoughts

What is a Leading Indicator?

Leading indicators are used to identify the potential for an accident before it occurs so measures can be taken to prevent it. They are based on the assumption that major accidents are not due to a unique set of random, proximal events. Instead, accidents result from

> *The migration of an organization to a state of increasing risk over time*
> *as safeguards and controls are relaxed*
> *due to conflicting goals and tradeoffs and reduced perceptions of risk*
> *leading to more risky behavior.*

This assumption implies that the march toward a major accident takes place over time and, therefore, the possibility exists to detect this dangerous path and intervene. A leading indicator is a signal that intervention is necessary.

[21] This chapter, like the others in this handbook, is written as a practical "how to" guide including some introductory explanatory information. For a more complete, technical treatment with lots of references, see Nancy G. Leveson, A systems approach to risk management through leading safety indicators, *Reliability Engineering and System Safety*, Elsevier Publishers, 2014.

For organizations, leading indicators can provide early indications about when aspects of the product, services, or organizational behavior are starting to go off track before the long-term effects on the organizational goals are visible.

What do People Use Now for Leading Indicators?

There has been a lot of effort put into identifying general leading indicators, particularly in the petrochemical industry. The attempts have involved trying to find a few generally applicable metrics that could raise flags in any company or even industry. Some examples are maintenance backlogs, minor incidents, equipment failure rates, or the results of surveys on employee culture (beliefs). The latter assumes that all or most accidents are caused by operators. Such attempts to identify leading indicators have not been found to be particularly effective. They have also mostly been directed at occupational safety rather than system safety.

Other attempts to identify leading indicators use incident and accident reports. One drawback is that only things that have happened before can be considered. Many of these previous factors have already been eliminated or controlled. An alternative approach is to use hazard analyses to predict future scenarios. A problem here is the limited nature of the scenarios generated by traditional hazard analysis techniques, which focus almost all attention on hardware failures.

Social, organizational, and managerial factors that lead to accidents have been hypothesized to be useful as leading indicators, but mostly only a small number (5 or 6) have been identified with little real data on how effective such general indicators are. The problem is that these hypotheses are not based on any model or framework on how and why accidents occur and therefore are very incomplete. In addition, these leading indicators only include the most common factors that are shared across a large number of organizations, which may omit the most important factors for a specific organization.

An alternative, which is described in this chapter, does not try to identify general leading indicators for all accidents but instead identifies leading indicators based on the design of safety controls within a particular company or even a specific product or service. We called these Assumption-Based Leading Indicators.

What are Assumption-Based Leading Indicators?

The idea of assumptions being the basis for identifying leading indicators was originally proposed and used for risk management programs outside of engineering. RAND developed the methodology of assumption-based planning (ABP) in the late 1980s to assist U.S. Army clients with mid- and long-term defense planning and to reduce uncertainty and manage risk in Army missions.

We have taken this basic idea and created an approach to risk management in engineering. This approach is based on the premise that *useful leading indicators of increasing risk can be identified based on the assumptions underlying the safety design process for the specific organization, product, or operations*. All engineering involves assumptions about the behavior of the operational system, even when that system is an organizational or managerial structure. Such assumptions may include, for example, failure rates for a hardware component over time, how a product will be used and the environment in which the product is used or the services are provided, the basic training for people on the tools they are tasked to use, or assumptions about the information needs for decision making and how effectively the information channels operate. Other external or environmental assumptions may arise from what it is believed that customers want, which could change over time as the whole marketplace changes.

The formal definition of an assumption-based indicator is therefore:

An assumptions-based leading indicator is defined as a warning sign that can be used in monitoring a process to detect when an assumption is broken or dangerously weak or when the validity of an assumption is changing.

The goal of a leading indicators program should be to monitor the social, managerial, and product/service assumptions made by the designers of the engineered system, both to find assumptions that originally were incorrect—such as beliefs about how people or other system components will behave in a certain context—as well as those that were originally correct but become incorrect over time as people optimize local goals or the environment changes. Instead of trying to find leading indicators that apply to *all* organizations and systems, leading indicators are generated for a specific organization or system design based on the safety-related assumptions about that specific design.

While there may be many reasons why an organizational structure may not operate the way it was designed to, one important cause is that the organizational culture, i.e., the goals and values of the organization and those in it,[22] does not conform to that which is required to achieve the program goals. Alternatively, the culture may degrade over time, perhaps influenced by competitive, financial, or other pressures and personnel changes or because of simple regression to familiar habits. This cultural drift needs to be detected and corrected as part of an effective leading indicators program.

Where do assumption-based leading indicators come from? In the most basic sense, they represent assumptions about how the system goals will be achieved. There are, in general, three types of assumptions involved:

- That the models and assumptions used in the initial design of the engineered system (including the organizational structure) were correct.
- That the system will be constructed and operated in the manner assumed by the designers.
- That the original models and assumptions are not violated by changes in the system over time, perhaps in an attempt to improve or optimize the processes, or by changes in the environment.

These types of assumptions are used in an assumptions-based leading indicator program to deciding what should be checked, how, when, and what actions should be taken if the checking determines an assumption is no longer true (or perhaps never was). Such a program has three aspects:

1. Identifying appropriate and effective leading indicators,
2. Creating a leading indicator monitoring program to use them, and
3. Embedding this monitoring system within a carefully designed risk management program. Detection alone, in the form of the first two aspects, is not enough—there must be a management process in place to act when the leading indicators show that action is necessary.

Each of these three aspects is now described.

Identifying Appropriate and Effective Leading Indicators using STAMP and STPA

Despite much effort to avoid them, accidents still occur. Theoretically, if we design a safe system, that is, eliminate or adequately control or mitigate all the hazards and nothing changes, then we should not have accidents. The problem is that neither of these conditions is usually true in practice: no engineering process is perfect nor is human behavior. In addition, every system and its environment are

[22] The chapter on designing a Safety Management System contains a more detailed definition and discussion of organizational culture.

subject to change over time. The starting point in seeking more effective leading indicators is to consider the general causes of accidents.

A General Categorization of the Causes of Accidents

The causes for accidents may arise in technical system development, in operations, and in management. Usually several or all of these types of causes can be found in accident scenarios. The following list describes the reasons accident causes occur in each of these three areas:

Design and Manufacturing

- Inadequate hazard analysis: Assumptions about the system hazards or the process used to identify them do not hold.
    - HA is not performed or is not completed
    - Some hazards are not identified due to inadequacies in the hazard analysis process or in how it is performed. Important causes are omitted and therefore not handled
    - Hazards are identified but they are not handled because they are assumed to be "sufficiently unlikely."
- Inadequate design of control and mitigation measures for the identified hazards, possibly due to inadequate engineering knowledge or to inappropriate assumptions about operations.
- Inadequate construction of the control and mitigation measures.

Operations

- Controls that designers assumed would exist during operations do not actually exist, are not used, or turn out to not be effective.
- Controls exist, are used, and originally were effective, but changes over time violate the assumptions underlying the original design of the controls.
    - New hazards arise with changing conditions, were not anticipated during development, or were dismissed as unlikely to occur
    - Physical controls and mitigation measures degrade over time in ways not accounted for in the analysis and design process
    - Components (including humans) behave differently over time (violate assumptions made during design and analysis)
    - The system environment changes over time (violates assumptions made during design and analysis)

Management

- The safety management system design is flawed (see the chapter on how to design an effective safety management system in this handbook).
- The safety management system could be effective but does not operate the way it was designed (and assumed) to operate. While there may be many reasons for misbehavior, one general cause is that the safety culture, i.e., the goals and values of the organization with respect to safety, degrades over time or was ineffective from the beginning. In addition, the behavior of those making safety-related decisions may be influenced by competitive, financial or other pressures.

To prevent accidents, we must eliminate or reduce the occurrence of these causes. A leading indicators program can be used to attempt to detect them before an accident occurs. Let's look at each of these three categories of accident causes.

During design and manufacturing, the hazard analysis process used may be inadequate or may be skipped. Sometimes, schedule pressures lead to skipping the hazard analysis process or only superficially performing one. Accidents occur frequently today because the hazard analysis techniques are incapable of identifying the new causes in today's complex, software-intensive systems. The elimination of these factors related to accident causation arising during system development and implementation may seem relatively simple on the surface, but may be quite difficult practically and politically within some organizations. Suggestions are provided in the chapter of this handbook on integrating STPA into organizations, but extensive discussion is beyond the current scope of this handbook.

A very common cause of accidents is that the technical factors were identified during development but ruled out because they were calculated to be sufficiently unlikely that they could be ignored. The best solution for this is not to use probabilistic risk assessment or informal likelihood assessments as a justification for ignoring some hazards or their causal scenarios. This solution may not be politically possible, but a leading indicator can be created that identifies when events happen that are incorrectly judged to be impossible or remote enough to ignore. In a recent chemical plant explosion and fire that I was analyzing, a catalyst was determined in the 1970s to be inert and thus the possibility of it contributing to an accident was judged to be zero. However, over many years the composition and manufacturing process for the catalyst changed. There were, in fact, two explosions related to this catalyst at chemical plants owned by this same company. Yet, when a new plant was designed by engineers at this company (after the two explosions), the designers still assumed that the catalyst was inert and ignored it in the risk assessment. That plant later had a serious accident related to the use the same catalyst. The original assumption clearly became incorrect over time. A leading indicator should have been triggered after the first of the two previous explosions, let alone after two of them, and the occurrence of the explosions should have led to changes in design assumptions.

As another case, the causes may be identified in the hazard analysis, but control and mitigation measures may be poorly designed and implemented. An example is inadequate design of physical control mechanisms. Simple calculation or knowledge errors may be involved, but incorrect assumptions can also play an important role. The common engineering design assumption about independence when redundancy is used to protect against failures is an example. Consider the Macondo (Deep Water Horizon) blowout preventer. There was redundancy in the means to control a potential blowout, but the redundant units contained a common cause failure mode. Acceptance of the belief that blowout preventers never failed was widespread in the industry despite the fact that ineffective blowout preventers had previously contributed to several serious accidents. A clear leading indicator that the assumption was wrong existed but was ignored. Challenger is another example of a common cause failure, only this time the assumption about the independence of the O-rings was checked and invalidated scientifically many years before the fatal *Challenger* flight. However, the change was never documented in the Marshall Space Center data base where the launch decision was made. Given the very large number of accidents that have involved common mode/cause failure, it appears to be an important engineering design assumption to revisit as are others that have contributed to many accidents. In this case, a flawed safety information system was also involved.

Sometimes the hazard controls that are assumed to be used during operations are poorly designed and ineffective, perhaps because of unidentified conflicts with other controls or incentives. They also may not be adequately implemented or used. As mentioned previously, the potential for migration toward violation of the operational assumptions that underpin the safety of the system needs to be reflected in the set of leading indicators. For example, operators may start to take shortcuts or turn off

safety devices in order to operate more efficiently; or, in an air traffic control system, the airspace may become more crowded than originally considered during system design. The assumptions that need to be checked here must be documented and passed to the operators.

Finally, assumptions involved in the design and operation of the safety management system (see Chapter 6) may be or become untrue. The goals and values of those participating in an industry or organization, i.e., the safety culture, is an important assumption that when wrong can be a major factor in accidents and must be reflected in the set of leading indicators. For example, a safety policy is a basic requirement for every company or organization to communicate the desired safety culture and behavior expected of individuals. There must be a way to measure how well that policy is being followed and if adherence decreases over time. Assumptions about management behavior and decision making are also commonly found to be violated after accidents occur, and these assumptions must be monitored.

How to Generate the Assumptions on which to Base Leading Indicators

Identifying appropriate and effective leading indicators can be accomplished, at least partially, using STPA. The goal is to use the hierarchical control structure model and the controls built into the organization to identify the assumptions on which constraints about the emergent property is based and to determine how to detect any weakening effectiveness of the controls designed to achieve the system goals. Additional assumptions can be identified during the system design process, as described in this section.

A list of where assumptions may come from is shown in the box below.

Assumptions may be generated from:

- High-level system goals, especially those including quantities, generated during concept development
- System-level requirements generated from the system goals
- Assumptions about the external environment in which system will operate
- System behavioral requirements imposed by safety-related environmental requirements and constraints (including constraints on the use of the system)
- STPA-generated hazards, the hierarchical control structure, unsafe control actions, and causal scenarios
- Design features devised to manage the causal scenarios
- Operational requirements created to manage causal scenarios that cannot be entirely handled through system design features
- Limitations in the design of safety-related controls, including operational controls
    - Limitations related to achievement of functional requirements
    - Limitations related to environmental assumptions
    - Limitations related to hazards and causal factors that could not be completely eliminated or controlled in the design or operational procedures
    - Limitations arising from tradeoffs made during system design

An example aircraft collision system called TCAS (Traffic alert and Collision Avoidance System) is used to illustrate the generation of critical assumptions on which leading indicators can be based. My

students and I assisted with the official certification of TCAS in the early 1990s, and later two of us created a TCAS intent specification.[23]

A critical part of an intent specification is the documentation of the assumptions under which the safety of the system is based. The examples here come from that specification and almost all are related to the scenarios generated by the qualitative hazard analysis[24] used for TCAS certification. A few of the assumptions did not come from the hazard analysis scenarios but instead from the design goals and the environment in which TCAS was expected to operate. Although STPA did not exist at the time that TCAS was designed and certified, we have since shown that STPA would generate all of the hazardous scenarios generated by the fault trees used plus many that were omitted from the fault tree analysis. Thus all the example assumptions below could be generated from the STPA results.

Generation of assumptions should begin during early concept development. Clues may be found when the high-level system goals and requirements contain numbers, but that is not necessary. In TCAS, two of the TCAS system goals are:

***G1**: Provide affordable and compatible collision avoidance system options for a broad spectrum of National Airspace System (NAS) users.*

***G2**: Detect potential midair collisions with other aircraft in all meteorological conditions; throughout navigable airspace, including airspace not covered by ATC primary or secondary radar systems, and in the absence of ground equipment.*

The system goals are used to create system requirements. Assumptions underlying these requirements can be generated at the same time. For example,

***1.18**: TCAS shall provide collision avoidance protection for any two aircraft closing horizontally at any rate up to 1200 knots and vertically up to 10,000 feet per minute [**G1**].*

Assumption: This requirement is derived from the assumption that commercial aircraft can operate up to 600 knots and 5000 feet per minute during vertical climb or controlled descent and therefore two planes can close horizontally up to 1200 knots and vertically up to 10,000 fpm.

This assumption is an example of something that will need to be checked in the future to ensure that technological changes have not contradicted it, making vulnerable all the technical design decisions that were based on it (which can be identified by the traceability pointers in an intent or other specification).

Another example system requirement is that:

***1.19.1**: TCAS shall operate in enroute and terminal areas with traffic densities up to 0.3 aircraft per square nautical miles (i.e., 24 aircraft within 5 nmi) [**G2**].*

Assumption: Traffic density may increase to this level by 1990, and this will be the maximum density over the next 20 years.

---

[23] *Intent specifications* are based on system theory, system engineering principles, and psychological research on human problem solving and how to enhance it. The goal is to assist humans in dealing with the complexity of specifying a complex system. Chapter 10 in *Engineering a Safer World* provides more information.

[24] A very extensive fault tree analysis—that included detailed human and software behavior—was used in the TCAS development and certification effort. Because the goal was completeness over quantification, no attempt was made to limit the fault tree boxes to things that could be quantified. The resulting fault tree was the most complete I have seen in my 35 years in safety engineering. It did not include all the scenarios that would be found by STPA, but it also did not exclude scenarios for which quantification was not possible which happens when the goal is to produce probabilities for the scenarios.

Again, future aircraft performance limits may change or there may be significant changes in airspace management, such as reduced vertical separation or very different ways of handling air traffic. Lots of computations in TCAS are based on the assumption underlying requirement 1.19.1, and it needs to be monitored to trigger re-evaluation of the safety parameters if it changes.

Another type of assumption may be identified (and specified) to explain a decision or to record fundamental information on which the design is based. For example, the design may be based on assumptions about the environment in which the system will operate. Examples from TCAS include:

EA1*: High-integrity communications exist among aircraft*

EA2*: The TCAS-equipped aircraft carries a Mode-S air traffic control transponder.*[25]

EA3*: All aircraft have operating transponders*

EA4*: All aircraft have legal identification numbers*

EA5*: Altitude information is available from intruding targets with a minimum precision of 100 feet.*

EA6*: The altimetry system that provides the aircraft's pressure altitude to the TCAS equipment will satisfy the requirements in RTCA Standard …*

EA7*: Threat aircraft will not make an abrupt maneuver that thwarts the TCAS escape maneuver.*

New technology and new types of aircraft integrated into controlled airspace could violate these assumptions. **EA4** is an example of a non-technical assumption. Identification numbers are usually provided by the aviation authorities in each country. That assumption will need to be ensured by international agreement and monitored by some international agency. The example assumption that aircraft have operating transponders (**EA3**) may be enforced by the airspace rules in a particular country and, again, must be ensured by some group. The truth of this assumption is critical as TCAS will not display any aircraft without an operating transponder nor provide a resolution advisory (RA).[26] **EA7** is an example of an assumption on the behavior of pilots and the air traffic control system and could also be violated by the introduction of unmanned or other types of new aircraft into the airspace.

Some assumptions may be imposed on the system by environmental requirements and constraints. Those assumptions may lead to restrictions on the use of the new system (which will require assumption checking) or may indicate the need for system safety and other analyses to determine the constraints that must be imposed on the system being created or on the larger encompassing system to ensure safety. Examples for TCAS include:

E1*: The behavior or interaction of non-TCAS equipment with TCAS must not degrade the performance of the TCAS equipment or the performance of the equipment with which TCAS interacts.*

E2*: Among the aircraft environmental alerts, the hierarchy shall be: Windshear has first priority, then the Ground Proximity Warning System (GPWS), then TCAS.*

E3*: The TCAS alerts and advisories must be independent of those using the master caution and warning system.*

These assumptions would only need to be checked when major design changes are introduced in the aircraft.

STPA will provide important input to the assumptions and leading indicator process, starting with the specification of the system hazards and proceeding through the entire modeling and analysis process. As an example for TCAS, the system hazards were identified as:

[25] An aircraft transponder sends information that assists air traffic control in maintaining aircraft separation.

[26] A resolution advisory is basically a maneuver that will avoid the intruder.

***H1**: TCAS causes or contributes to a near midair collision (NMAC), defined as a pair of controlled aircraft violating minimum separation standards.*

***H2**: TCAS causes or contributes to an aircraft coming too close to a fixed structure or natural terrain.*

***H3**: TCAS causes or contributes to the pilot losing control over the aircraft.*

***H4**: TCAS interferes with other safety-related aircraft systems (for example, ground proximity warning)*

***H5**: TCAS interferes with the ground-based air traffic control system (e.g., transponder transmissions to the ground or radar or radio services).*

***H6**: TCAS interferes with an ATC advisory that is safety-related (e.g., avoiding a restricted area or adverse weather conditions).*

The first basic set of safety-critical assumptions is that hazards will not occur in a properly designed and operated system. Any occurrence of one of these hazards (even if an accident does not result) should trigger a complete review of the safety engineering process, in this case, the process used to eliminate or mitigate TCAS hazards. Checking an assumption after the hazard has occurred is likely too late to prevent a loss, however, but the identification of hazards serves as a starting point from which earlier checks can be derived by using STPA to identify the scenarios that can lead to a hazard.

Additional assumptions can be deduced even at this high level, for example, that there is a ground-based air traffic control system (which could change in the future) and that TCAS will not interfere with its operation. While hazards rarely change, new ones may be introduced when changes are made to the system and the process used to handle them may be undermined.

Checks for the occurrence of hazards and unsafe control actions also provide important information about the adequacy of the hazard analysis process itself. The goal of hazard analysis and safety engineering is to identify hazards and then either eliminate or prevent them. If they cannot be prevented, then they need to be mitigated. Hazards that the engineers thought were eliminated or prevented should, of course, never occur. If they do, this event is an indication of flaws in the engineering process or perhaps in the assumptions made about the operational system, such as assumptions about pilot or air traffic controller behavior. It is not just enough to fix the technical process. The holes in the development process that allowed hazardous behavior to occur also need to be fixed.

Ideally, flaws in engineering practices or operational behavior assumptions will be identified by leading indicators before the actual hazardous states occur. This goal can be achieved by identifying the assumptions underlying the hazardous scenarios identified by STPA. The hierarchical control structure design, the UCAs, and the causal scenarios will be useful.

The control structure itself contains assumptions that can be used to create leading indicators. A midair collision over Uberlingen Germany in 2002 demonstrates the role of assumptions about the operation of the control structure in safety. There were three groups involved that had potential responsibilities over the pilot's response to a potential NMAC: TCAS, the ground air traffic controller, and the airline operations center. The latter provides the airline procedures for responding to TCAS alerts and trains the pilots in them. The relevance of the other two groups is obvious. Clearly, any potential conflicts and coordination problems between these three controllers will need to be resolved in the overall design and operation of the air traffic management system. In the case of TCAS, the RA provided by TCAS was assumed to always to be followed in the case of conflicting advisories. The designers decided that because there was no practical way, at that time, to downlink information to the ground controllers about any TCAS advisories that might have been issued to the crew, the pilot was to immediately implement the TCAS advisory and the co-pilot would transmit the TCAS alert information

by radio to the ground ATC so that the ground air traffic controller would know the state of the airspace and the advisories being given. The airline would provide the appropriate procedures and training to implement this protocol.

Several important assumptions about how conflicting advisories would be handled were violated in the Uberlinger midair collision. For example, there were supposed to be two air traffic controllers in the ground ATC tower, the pilots are supposed to follow the TCAS maneuver when there is a conflict between the advisory provided by the ground ATC system and TCAS, and the airline was assumed to be training pilots to follow the TCAS alert in such a conflict situation. The first of these assumptions had been violated for a while at night in the Swiss air traffic control center handling the two aircraft at the time of the tragedy. It is unknown whether the second one had been violated previously as that information was never checked. The third assumption, i.e., that the airline involved was training pilots to always follow TCAS when presented with conflicting advisories also had not been true for a long time, but nobody apparently was given the responsibility for ensuring that such training was occurring or they had not been exercising that responsibility. These incorrect assumptions about the operation of the control structure, if checked, could have served as leading indicators that the designed control structure was degrading.

The unsafe control actions, safety constraints, and causal scenarios also provide information about critical assumptions. The causal scenarios are used to design the controls and thus form the assumptions under which the controls are created. For example, **H5** gives rise to the following system safety constraint:

> ***SC.2****: TCAS must not interfere with the ground ATC system or other aircraft transmissions to the ground ATC system (**H5**).*

STPA can be used to identify causal scenarios for the violation of **SC.2**. This information can then be refined into a more detailed safety constraint **SC2.1**:

> ***SC2.1*** *The system design must not interfere with ground-based secondary surveillance radar, distance-measuring equipment channels, and with other radio services that operate in the 1030/1090 MHz frequency band (**2.5.1**).*

The assumption underlying a safe design of TCAS is that such interference will never occur. If it does, then this is a leading indicator that the design or operation of the system is flawed.

Humans tend to change their behavior over time and use automation in different ways than originally intended by the designers. Therefore, assumptions about operator behavior provide another important source for identifying leading indicators. For example, **H3** is that TCAS causes or contributes to the pilots losing control over the aircraft. Safety constraint **SC.6**, which is derived by STPA from **H3**, says:

> ***SC.6****: TCAS must not disrupt the pilot and ATC operations during critical phases of flight nor disrupt aircraft operation (**H3, 2.2.3, 2.19, 2.24.2**).*

Besides identifying the related hazard from which this safety constraint was derived (in this case **H3**), the specification here also points to features of the design (labeled **2.2.3**, **2.19**, and **2.24.2** in my TCAS intent specification) used to control that hazard, i.e., to enforce **SC.6**. These controls also contain important assumptions that need to be checked. The most basic assumption is that these controls will be effective in preventing the hazardous scenario and that they are implemented correctly. For example, in the STPA analysis, one of the scenarios identified that could lead to the violation of **SC.6** is that TCAS provides distracting resolution advisories while the pilot is on the ground or in the middle of taking off. A control was designed to prevent this scenario that allows the pilot to inhibit the production of potentially distracting resolution advisories during critical phases of takeoff and landing:

SC6.1 *The pilot of a TCAS-equipped aircraft must have the option to switch to the Traffic-Advisory-Mode-Only where traffic advisories are displayed but display of resolution advisories is inhibited (**2.2.3**).*

Assumption: This feature will be used only during takeoff or in final approach to parallel runways, when two aircraft are projected to come close to each other and TCAS would call for an evasive maneuver.

Addition of the control, i.e., the ability of the pilot to inhibit TCAS resolution advisories by switching to Traffic-Advisory-Only mode, creates another hazardous scenario that must be controlled through pilot procedures, training, etc., and leads to another assumption that should be checked during operation of the system to ensure that pilots are not violating the assumption associated with **SC6.1**.

Other examples of operational requirements that were created to eliminate or control hazardous scenarios are:

OP4*: After the threat is resolved, the pilot shall return promptly and smoothly to his/her previously assigned flight path.*

OP9*: The pilot must not maneuver on the basis of a Traffic Advisory only*

Because these procedures were created to counter specific scenarios that were identified as leading to hazards, they represent a source of assumptions that should be checked to identify hazardous behavior that could lead to an accident. Note that this information about violating assumptions could potentially be collected automatically using FOQA[27] systems.

As another example, in the Uberlingen accident, there were additional causal factors not mentioned earlier. One was that maintenance was being performed on the ATC equipment at the time of the collision, which disabled the air traffic controller's aural conflict alert without him knowing that it was not working. If the air traffic controller had known the alert was disabled, he could have adjusted his behavior. This type of causal factor can be controlled in operational procedures, in this case, the procedures for performing maintenance while the ATC tower is still operating. An additional important assumption, of course, is that such procedures are being followed and this assumption will also need to be checked.

A final source for assumptions that can be used to identify leading indicators is limitations in the design of safety-related controls. These limitations should be documented as they represent important information in the decision about whether and how the system should be deployed. Some TCAS limitations are related to the basic functional requirements, for example:

L2*: TCAS does not currently indicate horizontal escape maneuvers and therefore does not (and is not intended to) increase horizontal separation.*

Other limitations are related to the environmental assumptions, for example:

L1*. TCAS provides no protection against aircraft without transponders or with nonoperational transponders (**EA3**).*

L6*: Aircraft performance limitations constrain the magnitude of the escape maneuver that the flight crew can safely execute in response to a resolution advisory. It is possible for these limitations to preclude a successful resolution of the conflict (**H3, 2.38, 2.39**)*

[27] FOQA stands for Flight Operations Quality Assurance and is sometimes called Flight Data Monitoring (FDM). It refers to programs that collect and monitor data on an aircraft to improve flight safety and efficiency. Other industries have similar monitoring programs but use different names.

***L4**: TCAS is dependent on the accuracy of the threat aircraft's reported altitude. Separation assurance may be degraded by errors in intruder pressure altitude as reported by the transponder of the intruder aircraft (**EA5**)*

Assumption: This limitation holds for the airspace existing at the time of the initial TCAS deployment, where many aircraft use pressure altimeters rather than GPS. As more aircraft install GPS systems with greater accuracy than current pressure altimeters, this limitation will be reduced or eliminated.

An example assumption related to **L1** is that the operation of aircraft without transponders will be precluded in operations.

Limitations may relate to hazards or hazard causal factors that could not be completely eliminated or controlled in the design: Thus they represent accepted risks.

***L3**: TCAS will not issue an advisory if it is turned on or enabled to issue resolution advisories in the middle of a conflict.*[28]

An implied assumption here is that pilots will, except under unusual circumstances, turn TCAS on before taking off, which can be checked in performance audits.

Finally, limitations may be related to problems encountered or tradeoffs made during system design. For example, TCAS has a high-level, performance-monitoring requirement that led to the inclusion of a self-test function in the system design to determine whether TCAS is functioning correctly. The following system limitation relates to this self-test facility:

***L9**: Use by the pilot of the self-test function in flight will inhibit TCAS operation for up to 20 seconds depending upon the number of targets being tracked. The ATC transponder will not function during some portion of the self-test sequence.*

A safety-related assumption is that this behavior will be rare and therefore not result in frequent periods of non-operation of TCAS and therefore increased risk of an NMAC.

Using the Assumptions to Create an Assumption-Based Leading Indicator Program

The previous section described ways to generate assumption-based leading indicators. The assumptions must be maintained by implementing controls for the assumptions in the system design and checking leading indicators during operation of the system. How can the identified assumptions underlying safety be enforced? Some new terminology is useful in differentiating various cases.

First, in the original system design, actions to maintain assumptions, called *shaping actions*, may be taken to prevent the violation of the assumptions. More specifically, they are used to maintain assumptions, prevent hazards, and control migration to states of higher risk. Shaping actions provide feedforward control[29] over emergent properties and are built into the hierarchical control structure. Examples include physical interlocks to prevent entering a hazardous state; the use of desiccants to prevent corrosion; designing human operations to be easy and hard to omit; and creation of special management structures, such as an independent technical authority, for safety-critical decisions.

[28] The reasoning behind this limitation is beyond the scope of this handbook but arose from a scenario generated by the hazard analysis.

[29] Basic engineering terminology that may not be familiar to readers who are not trained as engineers is defined in Appendix F.

Hedging (contingency) actions are used to prepare for the possibility that an assumption will fail and take action accordingly. In STAMP terminology, hedging actions use feedback to control emergent system properties. In essence, they involve planning for monitoring the system behavior and responding to particular identified changes in the assumptions underlying the system controls. Such monitoring will include monitoring of the effectiveness of the shaping actions. An example is fail-safe design (e.g., protection and shutdown systems) that anticipates the potential for shaping actions to be unsuccessful in eliminating assumption violations in some cases.

Information for creating shaping and hedging actions can come from the STPA causal scenarios and the assumptions generated from these scenarios.

Checking assumptions need not be done continually. Some assumptions can only be violated in the presence of particular changes in the system or the environment. *Signposts* are points in the unfolding future where changes in the current controls (shaping and hedging actions) may be necessary or advisable. Signposts are points in time or specific future events or changes that trigger the checking of assumptions, including assumptions about the continued effectiveness of the controls. Management of change policies usually include various types of signposts. As an example, changes in the airspace, such as reduced separation or increased traffic density, might be identified as signposts for TCAS.

Shaping and hedging actions are designed into the safety control structure, both technical and organizational. Signposts are identified during the system design and development process and specific responses are created and specified. More general *assumption checking* during system operation involves specific checks to determine whether the assumptions underlying the design of the system are still valid. In assumption checking, risk managers and controllers monitor the system during its operation to determine whether the original assumptions are still valid. Such monitoring might include monitoring the response to signposts or perhaps changes and failures of assumptions that are known not to have been adequately or completely handled by shaping and hedging actions or have not been enforced at all by shaping and hedging actions. Checking might involve performance audits, surveys, and automatically collected FOQA or other data.

Some Ways to Enforce Assumption-Based Leading Indicators

- Shaping actions to prevent violation of the assumptions
- Hedging actions to prepare for failure of an assumption
- Assumption checking during operations
  - Sign posts to trigger specific checks
  - Checking during system operation (periodic or continual)
    - Performance audits
    - Surveys
    - Automatically collected data

In the previous chapter on organizational analysis, one of the conclusions of the example analysis was that technical independence was a desirable goal for the organization. In setting up such an independent technical authority, shaping actions need to be designed as well as feedback channels for the new controls. In addition, hedging actions might be established in case project managers simply ignore the independent authorities and make unwise decisions on their own. A third possibility, and one that is common, is that independence works well at the start, but it slowly is undermined by a series of small decisions or changes in circumstances over time. This latter case of degradation has recently been documented by the GAO for the NASA Independent Technical Authority established after the Columbia Space Shuttle loss. To find these problems early and not wait until near misses or worse occur, the

assumptions underlying the establishment of the independent authority (in this case) need to be documented and checked, either continually if feasible or periodically if not, to identify when the system is drifting to a higher state of risk through the degradation of the independent decision-making structure. The investigation of incidents and accidents are, of course, another place where failed assumptions can be identified, although it is rather late to do so after a loss.

Each leading indicator identified by STPA and other means should be documented by:

- The associated assumption(s),
- How it will be checked
- When it will be checked
- The action(s) to take if the indicator is true (the assumption is violated).

The actual structure of this documentation will depend on the organization and their risk management system.

Integrating Leading Indicators into Your Risk Management Program

Identifying leading indicators needs to be accompanied by the design of a risk management structure that can and will act when the leading indicators show that action is necessary. The leading indicators provide a foundation for risk management. The overall risk management process involves ensuring that not only are the risks initially handled in the new system design but also that leading indicators of failure of one or more of the assumptions and constraints are identified so that action can be taken.

General management principles apply in designing a risk management system that uses leading indicators, but some specific features are especially important. Even if the information is clear and available about elevated risk in the system, many organizations are still slow to process these signals and respond. Creating and monitoring leading indicators will not help if their appearance does not result in appropriate action being taken. Too often heuristic biases intervene, particularly defensive avoidance,[30] and clear leading indicators are ignored until too late to avoid serious negative consequences. In additional to psychological biases, organizational culture and politics can cause problems in designing and reacting to leading indicators.

Successful creation and use of leading indicators will require ways to control these psychological and cultural biases. For example, to encourage effective action, leading indicators have to be carefully integrated into the overall risk management program: Not only must they be communicated to appropriate decision makers, but detailed action plans for critical scenarios should be developed and triggers specified for implementing those action plans. Responsibilities need to be assigned for checking the existence of the leading indicators and for following through if problems are found.

As indicated in the previous section, every leading indicator should indicate when and how it will be checked and must have an action associated with it. Required action plans should be created *before* the assumptions are found to be invalid in order to lessen denial and avoidance behaviors and overcome organizational and cultural blinders. Responsibility for monitoring and action may need to be assigned to an independent organization and not to the program and project managers and those under

[30] Psychologists have written extensively about the biases humans show in assessing risk. These biases impact the way we design leading indicators and how we react to their occurrence. While all these biases are important, one of the most relevant in terms of reacting to leading indicators is called “defensive avoidance.” This type of bias may be reflected in the downgrading of the perceived accuracy of a leading indicator or reflected in people’s ability to take them seriously and to accept that risk may be increasing. Defensive avoidance is based on the common psychological tendency to rationalize and avoid consideration of a topic that is stressful or conflicts with other pressing goals.

conflicting pressures. Periodically, the list of leading indicators needs to be revisited and, if necessary, updated. A continuous improvement process should be created that reevaluates the current indicators over time in the light of experience and diagnoses any identified lack of effectiveness. For example, if a negative consequence occurs that was not signalled by leading indicators, an analysis should be made of why the leading indicators did not identify the problems in time to prevent the events or, if they did, why effective action was not taken. The entire leading indicators program should then be evaluated and improved using this information. Was a leading indicator identified but not checked? Or was an appropriate one never identified? Not only should the answer to these questions be used to improve the leading indicators themselves, but the reason for the omission or inadequate checking should be determined. That reason may involve a flaw in the process and lead to identifying other important programmatic assumptions that are not being checked.

Feasibility and Final Thoughts

Assuming that all of this sounds good, there is still the question of the feasibility of creating and checking leading indicators. While there are seemingly a very large number of potential leading indicators that result from the STPA process, not all of these need to be checked or to be checked frequently. Hedging and contingency actions are usually built into the safety control structure to control most of the identified causal factors. Many assumptions can be enforced by appropriate design, which should be the highest priority in dealing with the results of hazard analysis and is most feasible when hazard analysis is done in the early concept and design stages as described in the chapter on integrating STPA into system engineering.

Signposts will limit checking to the occurrence of certain triggering events. Documentation of critical assumptions should be part of any system development and specification activity. Documenting assumptions is not only important to identify leading indicators and improve safety, but also for the normal maintenance and evolution of the system. Even during initial development, people forget why they made certain decisions unless they are documented. In the case of our use of this approach to leading indicators on real problems, such as TCAS, we have found that documentation of critical assumptions are important to include in order to reduce the amount of development and testing time and to maintain the system without introducing hazards in the process. Note that I created the TCAS intent specification that included the safety-critical (and other) assumptions alone in my "spare time" over a period of a few months.

Could there still be a very large number of assumptions? Of course there may be. The usual approach to dealing with a large number of causal factors in safety is to try to reduce them by using probabilities. The problem is that there is no way to predict the probability of future events in any accurate way. Often the use of human (both engineering and management) expertise and knowledge is undervalued in comparison to a number, especially if that number is produced by a computer. But while numbers may appear more accurate, they are often only wild guesses based on little real data. Psychologists have found that, in fact, humans are very bad at estimating the probability of future events.[31] While certainly you are free to assign probability or likelihood to the leading indicators, we have found that risk levels based on engineering, management, psychological, and sociological knowledge produces better results. An additional consideration is that the independent review of non-numerical risk assessments is more likely to be "deeper" than numerical ones. People tend to believe that numbers are well founded and are less likely to challenge them unless they are wildly off the mark (and sometimes not even then).

[31] See, in particular, the work on *heuristic biases* by Tversky and Kahneman.

Chapter 7: Safety Management Systems

Nancy Leveson

The previous chapter describes how to use STAMP (systems thinking principles) and STPA to analyze a sociotechnical system. This chapter concentrates on the design and analysis of a specific type of sociotechnical system: the safety management system or SMS. Every organization creating or operating safety-critical systems has an SMS, just as every such organization has a safety culture, although the SMS and safety culture may be more or less helpful (and sometimes even harmful) in achieving societal and organizational safety goals.

This chapter describes how systems thinking (STAMP) and STPA can be used in creating a new SMS or in analyzing and improving an existing one. While STPA is mostly addressed indirectly here, a safety engineering program based on STPA will only be effective if it is used within a supportive management structure. The goal of this chapter is to assist in designing such a structure.

What is an SMS?

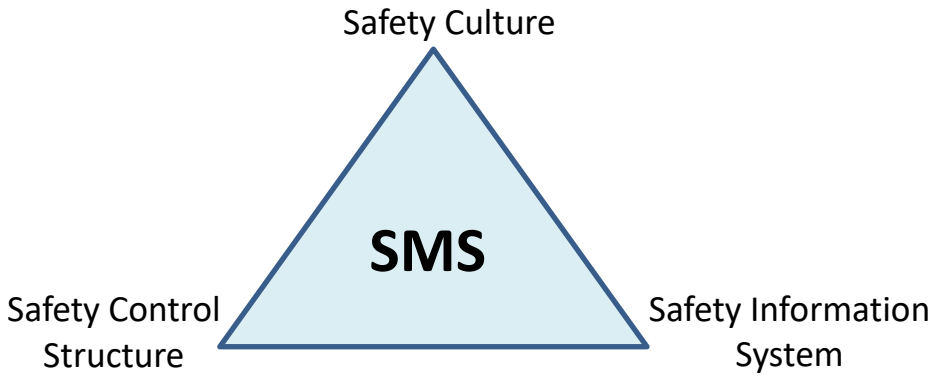
The goal of an SMS should be to proactively control (manage) safety in every aspect of the organization. For organizations that produce products, the SMS manages safety in the product development process and in the product itself. For service industries, safety must be designed into and managed in the workplace and the services provided. Because sociotechnical systems are rarely perfect from the beginning and the world changes over time, there must be an effective learning process in place that is continually learning from experience and improving the workplace and the products and services as well as the management system itself.

There is no one design of an effective safety management system. The goal of the design will depend on such factors as the type of organization (product development or service providers), the societal culture in which the organization resides and the organizational structure and culture, the inherent safety of the product or service, environmental factors, the other goals the organization wants to achieve and the regulatory environment. Some groups, such as the FAA, have specified one approach to risk management, but that may not be an appropriate solution for everyone, and risk management is only a subset of what is required in a safety management system. In addition, particular safety management system rules imply underlying cultural values that may not be the best ones for a particular group or organization in achieving high safety levels.

There are, however, some general characteristics of a successful SMS. The systems thinking approach to designing an SMS, described in this chapter, is the result of several decades of the author's experience in system safety engineering, in reading and writing accident reports and in identifying the factors involved, as well as helping organizations to prevent accidents. Some industries and organizations have lots of accidents, others have few, and some do not have accidents at all (e.g., SUBSAFE, the U.S. Navy nuclear submarine safety program[32]). Certain factors stand out as critical with respect to which of these three categories an organization or even an industry falls. Some of the most effective ways to control safety may not be practical for everyone. Designing an SMS is a system engineering problem: The goal should be to design and operate an SMS that is as effective as is feasible to implement in your organization.

[32] SUBSAFE focuses on two safety goals: (1) watertight integrity of the submarine's hull and (2) operability and integrity of critical systems to control and recover from a flooding emergency.

There are several components of the organization that are most important in managing safety: the culture, the safety control structure (management), and the safety information system. These three components cannot be considered in isolation, though. The systems approach suggests that these must all be part of a coherent and consistent whole in order to be most effective. Culture defines what desirable and acceptable behavior is and how decisions should be made. The management or control structure determines how that culture will be implemented in the organization. Finally, the safety information system provides the information necessary to make the management structure successful in achieving the desired safety culture: even the best of intentions will not suffice without the appropriate information to carry them out.



Safety Culture

People define safety culture in different ways. I like Edgar Shein's definition of safety culture:

Safety culture is the values and assumptions under which safety-related decisions are made.

Figure 6.1 shows Shein's three levels of organizational culture. The bottom level is the essence of the culture. The values and deep cultural assumptions form the basis for creating organizational rules, policies, and practices. These documents, in turn, describe how the surface level cultural artifacts (e.g., hazard analyses, accident investigation reports, etc.) are produced.

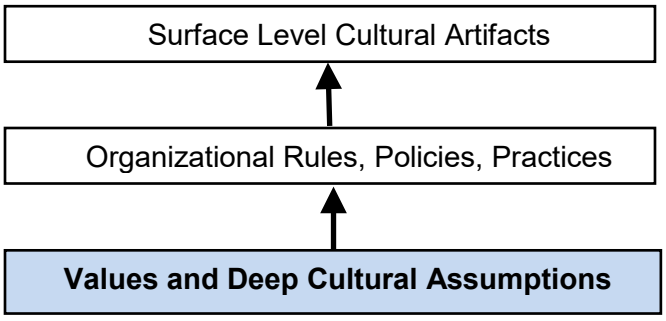


Figure 6.1: Shein's Three Levels of Organizational Culture

The middle and top levels are not the culture; they merely reflect the organizational culture. While attempting to make changes only at the top two levels may temporarily change behavior and even lower risk over the short term, superficial fixes at these levels that do not address the set of shared values and social norms on the bottom level are likely to be undone and become ineffective over time.

At the same time, trying to change culture without changing the environment in which it is embedded is also doomed to failure.

While sometimes people talk about "creating" a safety culture, there always exists some sort of safety culture, although the existing one may not be very useful in promoting safety. There is no such thing as a cultureless organization, industry, or society.

Often historical and environmental factors are key in creating the existing safety culture. For example, William Boeing, when building the commercial aviation industry, was faced with the fact that improved safety would be required to sell aircraft and air travel. In 1955, only 20% of the U.S. citizens were willing to fly—aircraft crashes were a relatively common occurrence compared with today. In contrast, the nuclear power industry was initiated by passage of the Price-Anderson Act in 1957 whereby they agreed to be tightly regulated by government in exchange for limited liability in case of accidents. In some industries, the safety culture was the result of strong individual leadership such as Admiral Hyman Rickover in the nuclear navy. Differences in the way safety is handled exist in each of these industries as a result.

While it is difficult to define a "good" safety culture vs. a "poor" one, there are some industries and organizations that have more accidents and major losses than others. The safety culture in organizations having relatively high accident rates tend to have one or more of the following characteristics:

- Culture of Risk Acceptance: This culture is based on the assumptions that accidents are inevitable and that nothing much can be done to prevent them beyond exhorting everyone to be careful. Accidents are considered to be the price of productivity. Often this assumption is accompanied by the belief that everyone should be responsible for safety—their own and others—and that accidents result from a lack of responsible behavior on the part of individuals. The belief is prevalent that if only everyone would act responsibly and safely, accidents would be reduced.
- Culture of Denial: In a culture of denial, risk assessment is often unrealistically low, with credible risks and warnings being dismissed without appropriate investigation: Management only wants to hear good news so that is what they are told. The focus is on showing the system is acceptably safe, not on identifying the ways it might be unsafe.
- Culture of Compliance: Focus here is on complying with government regulations. The underlying cultural belief is that complying with regulations will lead to acceptable results. Because regulatory agencies tend to focus on certifying that a product or service is safe, after the fact assurance is emphasized and often extensive "safety case" arguments are produced with little or no impact on the actual product or process safety.
- Paperwork Culture: This culture rests on the belief that producing lots of documentation and analysis paperwork leads to safe products and services. Piles of paper analyses are produced but they have little real impact on design and operations. The bulk of the safety-related paperwork may be produced by a group that is independent from and have little interaction with those who are designing and operating the products, implementing the processes, or providing the services.
- Culture of "Swagger": Safety is for sissies. Real men thrive on risk.

The features of a good safety culture are harder to specify, but they often include such things as: openness about safety and the safety goals, a willingness to hear bad news, an emphasis on doing what is necessary to increase safety rather than just complying with government regulations or producing a lot of paperwork, and believing that safety is important in achieving the organization's goals. In an effective safety culture, employees believe that managers can be trusted to hear their concerns about

safety and will take appropriate action, managers believe employees are worth listening to and are worthy of respect, and employees feel safe about reporting their concerns and feel their voice is valued. Safety is a shared responsibility where employees are part of the solution and not just considered the problem. At the same time, responsibility is not simply placed on the workforce to keep themselves and others safe.

How is the organizational safety culture established or changed? The safety culture (the values to be used in decision making) in any organization is set by the top management. A sincere commitment by management to safety is often cited as the most important factor in achieving it. Employees need to feel that they will be supported if they exhibit a reasonable concern for safety in their work and if they put safety ahead of other goals such as schedule and cost.

A manager's open concern for safety in everyday dealings with personnel can have a major impact on the reception given to safety issues. A classic story is about Paul O'Neill when he was hired from the outside as the CEO of Alcoa in 1989. He immediately announced that his primary goal was to make Alcoa the safest company in America and to go for zero injuries. Typical reactions to this announced goal were that the Alcoa board of directors had put a "crazy hippie in charge" and that he was going to destroy the company. In fact, within a year, Alcoa's profits had hit a record high and continued that way until O'Neill retired in the year 2000. At the same time, all that growth occurred while Alcoa became one of the safest companies in the world. O'Neill clearly understood that safety and productivity are not conflicting and, in fact, support each other.

In designing or engineering any system, goal setting and establishing the requirements for achieving those goals are the first step in any successful effort: If you do not know where you are going, you are unlikely to get there. Goals are necessary to guide any successful design process. Management establishes the value system under which decisions are made in an organization. Therefore, the first step in designing or improving the safety culture is for management to establish and communicate what is expected in the way of safety-related decision making and behavior.

Once the desired values have been determined by top management, the next step is to communicate the basic values of the leaders through a short written safety philosophy and more detailed safety policy to implement the general philosophy. Top management has responsibility to ensure that the written philosophy and policy has wide buy-in from the managers and employees.

Improving the safety culture, and thus safety, starts by the leaders communicating the type of safety culture they want to establish. Most companies have extensive safety policy documents that detail exactly how safety should be handled. While these are important, a shorter statement of the philosophical principles and values is a good way to communicate the expected cultural principles of the organization. This philosophical statement should define the relationship of safety to other organizational goals.

Some general cultural principles are applicable to all organizations, while the applicability of others will depend on whether the organization is developing safety-critical systems, operating them, or providing safety-related services. Some general principles that might be part of the Safety Philosophy statement:

1. All injuries and accidents are preventable.
2. Increasing quality and safety leads to decreasing cost and schedule and, in the long term, to increased profits. Preventing accidents is good business.
3. Safety and productivity go hand in hand. Improving safety management leads to improving other quality and performance factors. Maximum business performance requires safety.

4. Safety has to be built into a product or the design of a service. Adding it later will be less effective and more expensive. After-the-fact assurance cannot guarantee a safe design where safety is not already present. It is better to build safety in than try to ensure it after the fact.
5. The goal of accident/incident causality analysis is to determine why the loss (or near loss) occurred so that appropriate changes can be made rather than to find someone or something to blame.
6. Incidents and accidents are an important window into systems that are not operating safely and should trigger comprehensive causal analysis and improvement actions.
7. Safety information must be surfaced without fear. Safety analysis will be conducted without blame.
8. Safety commitment, openness and honesty is valued and rewarded in the organization
9. Effective communication and the sharing of information is essential to preventing losses.
10. Each employee will be evaluated on his or her performance and contribution to our safety efforts.

Number 4 requires some extra explanation as it violates how many organizations and even industries treat safety today. It is common to find an emphasis on measuring or assessing safety or risk after the fact rather than building safety into products and workplaces from the beginning. Designs are either safe or they are not, and all the arguing and assessing one can do will not change that fact. In addition, assessing or verifying after the fact is not only more expensive than designing safety in originally, it is also not very effective because the possible improvements are usually very limited after the design has been completed.

After-the-fact assessment can only provide confidence that safety already exists or evidence that it does not. Emphasis on after-the-fact assessment, therefore, leads to fudging numbers, often unconsciously, such as emphasizing only positive factors or those factors that will produce positive numbers (called *confirmation bias*), considering only factors that can be measured and can provide a number while ignoring other factors (such as management and design flaws), making up numbers or manipulating them so that numerical goals are achieved, etc. That does not mean that measurement and assessment are not important, just that at best it can only assure that an effort to create a safe system has been successful or not but it cannot make the system safe if it is not safe already.

An alternative to after-the-fact assessment is to go into the assessment process with a mindset to provide evidence that the design is not safe, rather than providing an argument that it is safe. This approach is more likely to be more effective. However, by the time this assessment occurs, it is usually too expensive to fix any problems that might be found or the fixes will be more costly and less effective than building safety into the design from the beginning. Starting the safety analysis process at the beginning of and during the design process will provide the best results (see the chapter in this handbook on integrating STPA into the system engineering process). That is, instead of proving safety, the emphasis should be on identifying and eliminating or controlling hazards. Those who emphasize after-the-fact assessment of safety are unlikely to achieve high safety.

In addition, an emphasis on probabilities or likelihood in assessment can lead to overconfidence and unrealistic assessment and to the omission of important factors that are not stochastic or for which probabilities cannot be obtained. In the very successful SUBSAFE program, all evidence used in certifying a system for safety must be Objective Quality Evidence (OQE). OQE is defined as “any statement of fact, either quantitative or qualitative, pertaining to the quality of a product or service based on observations, measurements, or tests that can be verified.” Probabilistic risk assessment (or even qualitative risk assessment that includes identifying the likelihood of a hazard leading to an accident) is an attempt to predict the future and cannot be verified beyond waiting many years to see if the

predictions were valid. Few effective crystal balls exist. Assessing likelihood is only effective when the past is identical to the future so past experience can be applied or it is close enough so that extrapolation from the past can be done. But new products are created to improve on past products in some way, not to duplicate them. And rare is the system (and its environment) that does not change over time. Even if the product itself does not change or degrade, people operating or using the system start to change their behavior, their uses of the system evolve, and the environment changes.

A written statement of the safety philosophy and more detailed policy statements are a start, but they are not enough. Employees quickly identify when the written policy differs from the actual behavior of managers. To be successful, there needs to be real commitment by those at the top, not just sloganeering and going through the motions.

How is commitment demonstrated? It is shown by setting priorities and following through on them; by personal involvement (e.g., top management chairing groups where safety decisions are made); by setting up appropriate organizational structures; by appointing designated, high-ranking leaders to safety-related responsibilities and providing adequate resources for them to be effective; by assigning the best employees to safety-related activities and rewarding them for their efforts; and by responding to initiatives by others. It is also communicated by minimizing blame. Leaders need to demonstrate that their highest priority is to fix the systemic factors leading to losses and not just find someone on which to place blame (usually someone at the lowest level in the organization as possible) and then moving on. Finally, leaders need to engineer the incentive structure in the organization to encourage the behavior desired.

The major ideas in this section are summarized in the following box:

Tips for how management can improve safety culture

- Set the goals and values to be used in decision making; establish and communicate what is expected in safety-related decision making and behavior.
- Support employees who exhibit reasonable concern for safety in their work
- Create a short written safety philosophy and more detailed safety policy
- Ensure safety philosophy and policy have wide buy-in from managers and employees
- Follow the safety philosophy in your decision making and expect the same from everyone
- Emphasize building safety in and not assurance or after-the-fact assessment
- Perform assessment with the goal of providing evidence that the design is not safe, not providing an argument that it is safe
- Require objective quality evidence for certification or assurance
- Demonstrate commitment to safety by
  - Personal involvement
  - Setting priorities and following through on them
  - Setting up appropriate organizational structures
  - Appointing high-ranking, respected leaders to safety-related roles and responsibilities
  - Providing adequate resources for safety-efforts to be effective
  - Assigning the best employees to safety-related activities, not just those who are nonessential or expendable
  - Rewarding employees for safety efforts
  - Responding to initiatives by others
- Minimize blame; focus on "why" not "who"
- Engineer the incentive structure to encourage desirable safety-related behavior

Safety Control Structure

Designing the safety control structure is just another form of system engineering, as discussed in the previous two chapters. It starts with a statement of the purpose of the effort. The SMS is:

What: A control structure that assists in creating and maintaining safety in an organization

Why: To ensure that hazards are eliminated or, if not possible, controlled (mitigated) and to promote an effective safety culture

How: By establishing management controls and responsibilities (RAAs) to manage hazards and a comprehensive and usable safety information system.

The overall goal is to design a control structure that eliminates or reduces losses. Satisfying this goal requires a clear definition of expectations, responsibilities, authority, and accountability for safety tasks at all levels of the control structure. In addition, to operate effectively the structure requires appropriate feedback and coordination between entities. It should also involve leading indicators to signal when the controls are becoming ineffective because of internal or external changes. Together the entire control structure must enforce the safety constraints on system behavior through physical design, defined processes and procedures, and social interactions and culture.

There will probably be significant differences between a safety control structure for development and one for operations. If you are a regulated industry, government regulatory agencies will need to be

included in your safety control structure model. Other external groups may be important to include such as the courts, insurance companies, user groups, etc.

General Safety Control Structure Design Considerations

Some basic considerations when designing of evaluating a safety control structure is how to assign responsibility for safety, the appropriate place(s) in the organization for safety-related activities, communication of information and coordination of activities, managing and controlling for change, designing and encouraging feedback, and determining the design and role of the risk management procedures to be used. In addition, there must be consideration of education and training efforts that will be required and how learning and continual improvement of the control structure itself will be assured. In addition, see Appendix D "Guidelines for Designing and Evaluating an SMS."

Assigning Responsibility:

Leadership is the key to achieving high levels of safety. That means that leadership of the organizational safety functions should not just be a rotational assignment for training future leaders and managers. Organizations that have few losses appoint leaders of the safety functions who are passionate about safety and the role they play in preventing losses. There should be a career path within the organization that allows those who are committed to preventing losses to rise in the safety management organization.

As with any effective management system, there must be responsibility, authority, and accountability assigned. A belief that "everyone is responsible for their own and others safety as well as the safety of the products" leads to excessive accidents. If everyone is responsible for safety, then nobody is.

Responsibility for safety lies at every level of the organizational structure although appropriate responsibilities will differ at all of them. In some, usually high-accident organizations, a belief is prevalent that responsibility for safety should be pushed down in the control structure. The argument is used that the lower levels have more information about how their parts of the system actually work. The problem with this argument is that the lower levels also lack perspective: Although they have detailed information about their specific parts of the system, they do not have information about other parts of the system and therefore cannot anticipate or prevent unsafe interactions among components of the entire system. Lower levels also tend to have a short-term focus rather than a longer-term one while the higher organizational levels have a broader focus on system goals as opposed to the goals of their own subcomponent of the system. The higher organizational levels need to control interactions among the lower level components and ensure that safety constraints are being enforced by those beneath them.

Each level must provide oversight of the level below by ensuring they are using appropriate procedures, carrying them out correctly, and that they are effective. There also usually needs to be a management focal point (or points) with responsibility for ensuring that the safety management system is designed and working properly at each level of the management system.

Tips for assigning responsibility

- Appoint leaders who are passionate about safety
- Create career paths for those committed to preventing losses
- Provide a clear definition of expectations, responsibilities, authority and accountability at all levels of the safety control structure
- Do not make everyone responsible for safety
- Assign safety responsibilities throughout and at all levels of the control structure
- Assign someone to be responsible for ensuring the SMS is designed and working properly

Place in the organization:

There are some basic principles that need to be considered in deciding where to place system safety activities in the organization, and how to design and manage feedback and control change.

First, there needs to be a high-level group with enterprise-level responsibilities such as ensuring that required activities are taking place and that they are effective. This group also provides leadership and coordination. Although safety activities will permeate every part of the development and operation of large organizations, a common methodology and approach will strengthen the individual activities. The leader of this group needs to have the ability to communicate directly with top management and provide input to all types of management decision making. This implies that the person in this position must report to someone with influence and be seen as having the support of senior management.

Below this top-level safety management, there may be activities at all levels of the organization, with the higher levels having broader responsibilities and each successive lower level having more focused responsibilities appropriate to the level at which they operate, for example, at the business unit, program and project level of a development organization. A common mistake is to make safety a staff function that has little impact on line operations.

There is no one or even several correct designs for a safety control structure. An effective design will depend on the industry, organizational management style, etc. Some responsibilities that should be included in the safety control structure for development and operations are listed in an appendix of this handbook.

Second, as has been mentioned before in this handbook, separating system safety engineering from system engineering is a mistake in product development organizations; it can lead to higher costs in achieving safety goals because safety concerns must be addressed starting early in the development process. Safety engineering efforts should not be focused on after-the-fact assurance. Service organizations have different concerns, but there should be some safety engineering component within all departments or groups where decision making about safety takes place. At the same time, there are reasons for also having independent oversight and decision-making authority outside the primary decision-making group. For all types of organizations, safety needs to be designed into the workplace and decisions need to be made about the appropriate place for this responsibility in the organization.

Some general principles for allocating safety responsibilities in the organizational structure are:

- Decision makers need direct links to those who can provide safety information. If critical information has to float up a chain of command, it may be lost or modified either deliberately, usually because of schedule or budget pressures, or inadvertently. Direct communication channels provide more chance that information is received in a timely manner and without

being changed by being filtered through groups with potentially conflicting interests. Some decision makers may also need fast access to information.

- Direct communication channels to most parts of the organization are required. Safety may be involved in almost all organizational activities.
- Safety must have influence on decision making, which means that decision makers have access to necessary safety information at the time safety-related decisions need to be made.

The SUBSAFE program for safety in U.S. nuclear submarines uses a unique design that satisfies many of these requirements. They describe their structure as a separation of powers or a "three-legged stool" (Figure 6.2). Managers can select only from a set of acceptable options (with respect to safety) derived by the Independent Technical Authority (ITA). Technical Authority is defined as a process that establishes and assures adherence to technical standards and policy. The ITA provides a range of technically acceptable alternatives with risk[33] and value assessments. Responsibilities (and accountability) include:

- Setting and enforcing technical standards
- Maintaining subject matter expertise
- Assuring safe and reliable operations
- Ensuring effective and efficient systems engineering
- Making unbiased independent technical decisions
- Providing stewardship of technical and engineering capabilities.

The third leg of the stool is the compliance verification organization. It is equal in authority to the program managers and the ITA.

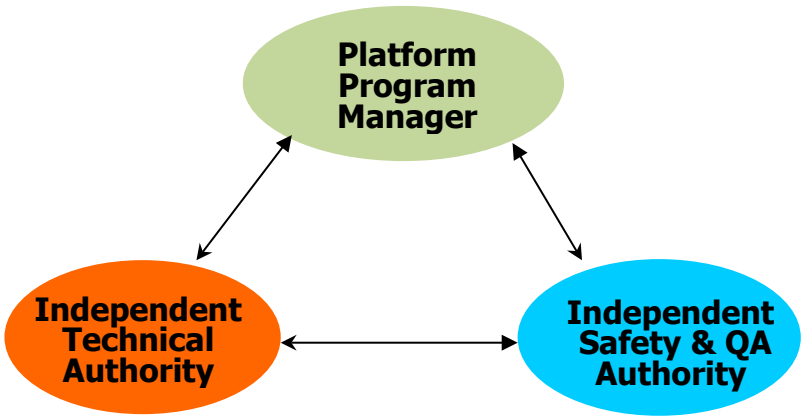


Figure 6.2: The SUBSAFE 3-Legged Stool Concept

[33] Risk may be specified quantitatively or qualitatively but, as explained elsewhere in this chapter, in SUBSAFE it must use objective quality evidence and therefore probabilistic risk assessment or likelihood estimates that cannot be tested or verified are not allowed.

Tips for where to locate activities in the control structure

- Create a high-level group with enterprise level responsibilities, such as ensuring required activities are taking place and they are effective and providing leadership and coordination.
    - This group ensures a common methodology and approach is being used by everyone
    - They must have a direct path to top management and provide input to management decision making
    - They must report to someone with influence and be seen as having the support of senior management
- Assign responsibilities so safety is not just a staff function but has direct impact and influence on line operations
- Do not separate system safety and system engineering in product development organizations. Integrate the safety function into the engineering function.
- Include safety engineering components in all departments or groups where safety-related decision making occurs.
- Provide independent oversight in addition to integrated functions
- Ensure decision makers have direct links to those who can provide safety analysis and information.
- Create direct communication channels between any parts of the organization where there are safety-related activities and safety decisions need to be made.
- Consider implementing the SUBSAFE separation of powers structure. At the least, ensure that technical decisions are independent of programmatic ones.

Communication and Coordination:

Feedback and dissemination of information to those who need it to perform their safety management roles is an important factor to consider in the design of your safety control structure. It is not just a matter of collecting lots (sometimes enormous amounts) of information. In fact, collecting too much information may degrade the information system when it overwhelms the ability to identify the information each person actually needs to make effective safety decisions. In addition, required information for improved safety-related decision making must be handy, that is, available when needed. Because it is so important in reducing losses, the safety information system is discussed separately later in this chapter.

Communication is also important in coordination of activities and responses to events. People with overlapping responsibilities need communication channels and ways to coordinate their activities to ensure that the safety constraints are enforced. For example, safety-motivated changes in one subsystem may affect another subsystem and the system as a whole. Safety activities must not end up fragmented and uncoordinated. Interactions must be defined not just between hierarchical components but also between different parts or types of systems at the same level, such as between development and manufacturing or between development and operations. Figure 6.3 shows an example of the types of safety-related information that needs to be communicated between system developers and system operators.

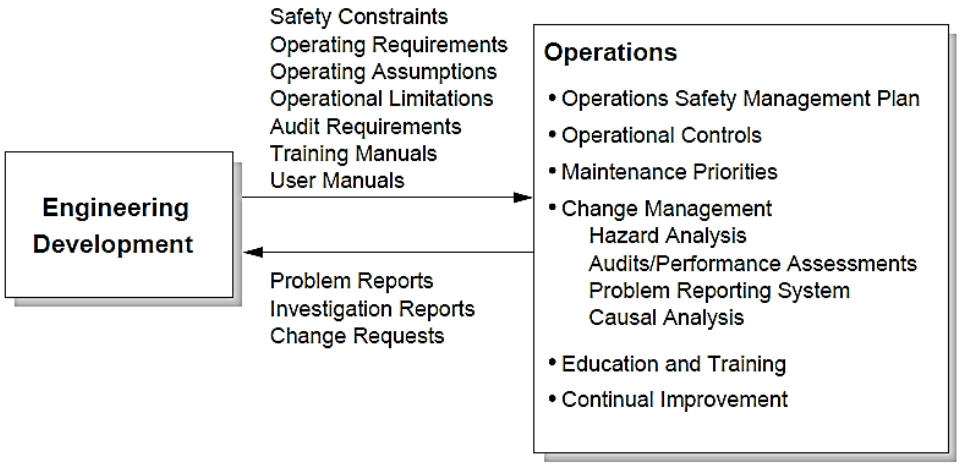


Figure 6.3. Necessary information channels between development and operations to support safety activities in both.

Similar descriptions of the information needing to be communicated among all the components of the safety control structure need to be identified and documented.

One very effective means for communication and coordination devised by the defense department is *working groups*. Defense department projects can span many years, be extremely large and complex, and often involve a large number of participants who are geographically and organizationally distributed. Coordination and communication can become a major problem in such projects. Part of the solution is provided by a hierarchical structure where the high-level management of the project may lie in the Defense Department, but there is a Prime Contractor that provides the system engineering and coordination among the subcontractors. For such a structure to work, communication becomes critical. Working groups have been successful in such coordination and communication efforts and can be adapted for less complex projects.

A working group provides an interface between two hierarchical components of the safety control structure or between two or more components at the same level. Members of these groups are responsible for coordinating efforts, reporting the status of unresolved safety issues, and sharing information about independent safety efforts. There may be working groups at each level of the control structure, with their members and responsibilities depending on the level at which they operate. A corporate working group may be composed of the safety managers for different divisions or programs while at the lower levels working groups may be composed of representatives from groups designing different parts of a product. When there are special types of safety problems that require new solutions, a safety group may be created to share experiences and approaches, such as a software safety working group. A government agency might create a safety working group for a large project composed of representatives from the agency offices and the prime contractor. A prime contractor may create a safety working group composed of the safety managers for all the subcontractors.

Tips for designing communication and coordination

- Make sure information necessary for decision making involving safety is available to decision makers.
- Provide communication channels and a way to coordinate activities among those with overlapping safety responsibilities
- Make sure safety activities are not fragmented and uncoordinated.
- Define required interactions and ensure that necessary information flow among them is defined and used
- Identify and document necessary safety-related communication channels among all components of the safety control structure.
- Ensure feedback and coordination channels exist and are working
- Consider establishing safety working groups

Managing and controlling change

Most accidents occur after some type of change.[34] This fact is not surprising as continuing to do what one has done without any changes in behavior or the environment should theoretically result in the same consequences. Adaptation and change is an inherent part of any system and is required for an organization to thrive. The problem is not change, but *unsafe* change. The SMS, therefore, must have carefully designed controls to prevent unsafe changes and to detect them if they occur despite the efforts to prevent them. The goal is to allow change as long as it does not violate the safety constraints. Two types of changes need to be considered: planned changes and unplanned changes.

Planned Changes: The SMS must continue to be effective in the face of planned changes, including changes in the organization, in human behavior and processes, in operations (including changes to maintenance phases and back to operational phases), or in the environment. Most companies have (and, if not, they should have) Management of Change (MOC) policies that require all planned changes to be evaluated with respect to their impact on safety. The cost of such evaluations will depend on the scope of the change, the quality of documentation, and how the original hazard analysis was performed. STPA incorporates complete traceability from identified hazards to their causal scenarios and to the design features that are used to prevent them. Without such traceability, the cost of reanalysis after a change may be impractical and therefore skipped.

In fact, MOC procedures are often skipped in practice, resulting in losses that need never have occurred. The safety control structure must have assigned responsibility for enforcement of MOC procedures and feedback channels to determine whether they are being followed and, if not, why. If the cause of skipping them is in the MOC change procedures themselves (too onerous, too difficult, too time consuming, etc.), then the MOC procedures may need to be changed.

The cost of implementing the MOC procedures will depend on the quality of the system documentation and how the original hazard analysis was done. A design goal for STPA was minimization of the cost in evaluating changes.

Unplanned changes: Unplanned changes present more difficult challenges. There needs to be (1) a way to identify potentially unsafe changes and (2) ways to respond to these changes. Leading indicators, as described in the previous chapter, should be created from assumptions made in the system (including the SMS) design and safety analysis process. Identifying leading indicators requires recording

[34] See Nancy G. Leveson, *Safeware*, Addison-Wesley Publishers for references.

assumptions during design and development of the organizational structure, the SMS, and the products and workplace. STPA control structures, UCAs, and causal scenarios provide information necessary to create effective leading indicators. If the causal scenarios cannot be eliminated, information from the STPA analysis can be used to create shaping and hedging actions as well as to identify signposts and assumptions to check in audits and performance assessments.

A common reason for unsafe changes to occur is risk re-evaluation. After a period of few or no losses, people begin to re-evaluate their view of the risks downward. Nothing may have changed so risk has not really decreased, but people may believe it has. Under pressure, they start to violate their own rules and justify it by arguing that safety will not be affected. The processes in the safety control structure need to interrupt this risk re-evaluation process before safety margins are eroded. One requirement is the communication of appropriate information about the actual level of risk. Management needs to be aware of the state of risk in the processes they are controlling. That requires appropriate feedback about the state of the designed safety controls.

In addition, there needs to be an alerting function to a person with responsibility when behavior is contrary to the true level of risk. The key to preventing this phenomenon is to allow flexibility in how safety goals are achieved and to provide information that allows accurate risk assessment by decision makers.

An effective SMS is characterized by continual vigilance against degradation of the system over time. The SUBSAFE program, as an example, puts a lot of effort into combating what they describe are their three most difficult challenges:

1. Ignorance: Not knowing;
2. Arrogance: Pride, self-importance, conceit, or assumption of intellectual superiority and presumption of knowledge that is not supported by fact;
3. Complacency: Satisfaction with one's accomplishments accompanied by a lack of awareness of actual dangers or deficiencies.

Much of the design and focus of the SUBSAFE program is aimed at combating these three challenges, and the design of any SMS should include steps to provide vigilance against them.

Tips for managing and controlling change

- Design controls and MOC policy to prevent unsafe changes and detect if they occur
- Evaluate all planned changes, including temporary ones, for their potential impact on safety
- Assign responsibility for ensuring MOC procedures are enforced and are being followed. If they are not, find out why and fix the problems.
- Create documentation and procedures that minimize the cost of performing the MOC procedures.
- Create ways to identify unplanned changes that could be unsafe ones and to respond to these changes.
    - Devise assumption-based leading indicators and create a risk management program that effectively monitors and responds when potentially unsafe changes are identified.
    - Record assumptions during design and development of the organizational structure, the SMS, the products, and the workplace.
    - Create shaping and hedging actions and a leading indicator checking program including audits and performance checking as well as signposts
    - Implement leading indicators to signal when controls are becoming ineffective
- Ensure that decision makers have information about the current level of risk and state of the designed safety controls
- Assign responsibility to respond when feedback shows that behavior does not match the true level of risk.
- Remain vigilant against the degradation of the safety control structure and the safety culture over time and any increase in complacency.

Designing and Encouraging Feedback

Accurate risk assessment requires information flow and properly functioning feedback channels. Cultural problems are a common reason for problems in feedback. There are three general ways to implement feedback channels: audits and performance assessments, accident/incident causal analysis, and reporting systems.

Audits and Performance Assessments: There are many types of audits and performance assessments, but those whose goal is to evaluate the state of safety start from the safety constraints and the assumptions in the design of the safety controls. Audits should involve not just the products and processes but also the safety management system itself and the effectiveness of the controls designed to ensure that losses are prevented. At least some part of a safety audit and performance assessment should be focused on the operation of the safety control structure as it was designed and assumed it would operate.

The entire safety control structure must be audited and not just the lower levels. In the SUBSAFE program, even the top Admirals are subject to a SUBSAFE audit. Not only does this ensure that the program is operating as assumed, but it also provides a positive cultural component to the audit in that all employees see that even the leaders are expected to follow the SUBSAFE rules and that top management are willing to accept and resolve audit findings just like any other member of the SUBSAFE community. People at lower levels of the SUBSAFE safety control structure participate in the performance assessment of those above them, providing a visible sign of the commitment of the entire program to safety and the importance of everyone in preventing losses. Accepting being audited and

implementing improvements as a result, that is, leading by example, is a powerful way for leaders to convey their commitment to safety and to its improvement.

Participatory and non-punitive audits can be very effective. The goal of such an audit is that it be a constructive learning experience and not a judgmental process. It should be viewed as a chance to improve safety rather than a way to evaluate employees. Instead of the usual observation-only audit process by outside audit companies, experts from other parts of the organization not directly being audited should make up the audit team. Various stakeholders may play a role in the audit and even individuals from the group being audited may be on the audit team. The goal should be to create an attitude that this is a chance to improve our practices and provide a learning activity for everyone involved, including the auditors.

The audit itself should involve continuous communication with those being audited so as to obtain full understanding of any identified problems and potential solutions. Unlike the usual rules for outside audits, in a participatory audit immediate feedback should be provided and solutions discussed. Doing this will reinforce the understanding that the goal is to improve safety and not to punish or evaluate those involved. It also provides an opportunity to solve problems when a knowledgeable team is on the spot and not after the usual written report is provided months after the audit actually occurs.

An important goal of safety audits and performance assessments is to measure the level of safety knowledge and training that actually exists, not what managers think exists or what exists in the training programs and user manuals. The audits can provide important feedback about potential improvement to the training and education activities. In keeping with the non-punitive audit philosophy, knowledge assessments should not be used in a negative way that is viewed as punishment by those being assessed. Instead, the goal should be to improve training and education efforts.

Incident and Accident Investigation: Incident and accident investigation provides clear evidence that the SMS is not working as designed or expected. The investigation procedures must be embedded in an organizational structure that allows exploitation of the results. All systemic factors involved must be identified and not just the symptoms or technical factors. Assigning blame should not be the goal; rather the goal should be to find out why the safety control structure was not effective in preventing the loss or near loss. That means that the entire safety control structure must be investigated to identify each component's potential contribution to the adverse events.

Managers should not be responsible for investigating incidents that occur in their chain of command: investigators and causal analysts must be managerial and financially independent from those in the immediate management structure involved. Using trained teams with independent budgets and with high-level and independent management should be considered.

Investigation requires more than just writing a report. There must be assignment of responsibility for ensuring that appropriate measures are taken to strengthen the aspects of the safety control structure that contributed to the events. Then there should be follow-up to ensure that the fixes were effective. Too often, fixes are made but there is no attempt to determine whether the fixes were successful in improving safety management. Finally, the findings should be used as input to future audits and performance assessments. If there is a reoccurrence of the same factors that led to incidents and accidents in the past, there needs to be an investigation of why those factors were never corrected or why they reoccurred even if they were removed for a while. If fixes are not effective in removing the causes of incidents, then an investigation of the process of creating recommendations and responding to them is warranted to identify any weaknesses in these processes in the organization and to improve them. That is, not only must the factors involved in the incident be corrected but also the process that led to inadequate fixes being implemented after previous incidents or accidents.

Reporting Systems: Reporting systems are critical. Often, after an accident, it is found that the same events occurred multiple times in the past but were never reported or, if reported, were never corrected. These events may involve near misses. For example, several aircraft may not fly an approach correctly, but because it did not result in an accident they may not report it, even if a reporting system exists. Not until an accident occurs is action taken.

A common finding in accident reports is that a reporting system existed but was not used. These reports then usually include recommendations to train people on how to use the reporting system and to require that they use it. This recommendation assumes that the problem is with the potential reporter and not with the design of the reporting system itself.

Examination of the reporting system itself may find that it is difficult or awkward to use and that information reported appears to go into a black hole. People may believe that there is no point in going through the official reporting system because the organization will not do anything anyway about the factors reported. Often, those finding problems bypass the reporting system and simply go directly to a person or group they believe may be able to solve the problems. While this might be effective and efficient in the short term, it may lead to the same problems occurring in the future because the systemic causes are not eliminated. In some cases, reporting is inhibited by a fear that the information reported will be used against the reporter. Anonymous reporting systems can be helpful here.

Another factor to consider is that often events are not reported by front-line operators because they identified the problem before it created a perception of risk or they may have perceived it as only their own error. Most people are not trained to recognize risk when it is created as part of a normal job process. People accept the flaws in design as “normal” and perhaps already known so rather than reporting them, they just work with or around them. A related factor is that people will generally not report hazardous events when those events do not meet the criteria for required reporting. Near misses may fall in the latter category.

In general, reporting needs to be encouraged. This can be accomplished by maximizing accessibility, minimizing anxiety, and acting on the information obtained. Reporting forms or channels should be easily and ubiquitously available and not cumbersome to fill in or use. To minimize anxiety, there should be a written policy on what the reporting process is; the consequences of reporting; and the rights, privileges, protections, and obligations of those doing both the reporting and those following up on the reports. Without a written policy, ambiguity exists and people will disclose less. Alternatively, ambiguity about who is responsible for following up may lead to everyone assuming that someone else will take care of it.

Finally, encouraging reporting involves providing feedback. Immediately after a report is created, the person who provides the information should be informed that the report was received, assured that it will be investigated, and thanked for their input. A second crucial component is providing feedback later about the results of any investigation and any steps that were taken as a result to prevent a reoccurrence. Reporters should not feel like their concerns are being ignored.

Tips for designing and encouraging feedback

- Design and ensure the continued efficacy of audits, performance assessments, and reporting systems.
- Audit the safety control structure itself (including all levels) and the effectiveness of the designed controls.
- Design audits so they are constructive learning experiences and not a judgmental process.
    - Include as participants members of the groups that are being audited.
    - Use audits as a way to improve safety and as a learning activity and not to evaluate employees.
- Take advantage of audits to evaluate the effectiveness of training and education activities and use them to provide feedback (knowledge assessment) to be used for improving training activities.
- Create effective system-level accident/incident causal analysis procedures that focus on *why* and not *who***.**
- Create incident and investigation procedures based on STAMP and systems thinking (such as CAST) that identify systemic factors and not just the symptoms of the deeper problems.
- Embed the investigation procedures in an organizational structure that allows exploitation of the results. Assigning blame or finding a "root cause" should not be the goal.
- Accident investigation should be managerially and financially from those in the immediate management structure involved. Consider using highly trained teams with independent budgets and high-level management. Follow up on recommendations to determine whether they were effective and, if not, then why.
- Ensure reporting systems are easy to use and available and anonymous reporting channels exist.
    - Encourage reporting and train people to know when it should be used.
    - Provide a written policy
    - Maximize accessibility, minimize anxiety, and act on information obtained.
    - Provide feedback to those using the reporting channels. Reporters need to feel like their concerns are not being ignored.

Risk Management

Sometimes the risk management system is called the "safety management system," but they are not the same. The SMS is a larger concept, that may (and usually does) involve risk management activities. Risk management is the set of activities associated with identifying hazards, analyzing them, and using this information to reduce losses through the design of products, processes, services, and workplaces. These activities will be embedded somewhere in the safety control structure. While risk management is comprised of the technical activities involved in preventing hazards, the SMS also contains organizational, managerial, and cultural aspects of safety.

STAMP implies a broader or at least different definition of risk. Risk has traditionally been defined as the severity and likelihood of hazards or accidents occurring. In contrast, in STAMP

> *Risk is defined in terms of the effectiveness of the controls used to enforce safe system behavior, i.e., the design and operation of the safety control structure.*

Note that this definition does not require the determination of likelihood of the events occurring but rather an evaluation of the effectiveness of the controls being used to prevent them.

Creating a risk management system involves designing the procedures to be used in performing the technical risk management activities and assigning responsibility for implementing these procedures to components of the safety control. Risk management may also involve creating leading indicators to identify when risk is increasing.

In fact, an important factor in the effectiveness of the risk management system lies in being able to identify when risk is increasing before a major loss occurs. All safety efforts involve assumptions about how the components of the system will behave and about the environment in which they operate. Violations of these assumptions will undermine the original risk identification and management assumptions. Virtually all accidents have precursors that were unrecognized or for which the responses were non-existent or ineffective. Precursors to accidents are incidents or conditions in which losses do not occur but could have under other circumstances. Leading indicators are characteristics of an organization or an organization's operation that indicate the safety management system is not operating as was assumed when it was designed.

Another way of looking at this is that leading indicators are simply evidence before losses occur that the assumptions under which safety was assured were flawed or are no longer true and that risk may be increasing. As an example, before the Shell Moerdijk chemical plant explosion in the Netherlands, there had been two instances where the same conditions led to explosions in other Shell plants. These explosions were never thoroughly explored or at least the results of the analyses were not used in later design activities, such as the design of the Shell Moerdijk plant where the assumption was made that the conditions were impossible and could not lead to an explosion. Surprisingly, probabilistic risk assessments are often not re-evaluated even after concrete evidence is gathered from actual use of the system that the assumptions made in the analysis are not true. Apparently, the power of a calculated number often trumps evidence that the number is untrue. Generating evidence about the truth of risk assessments and the assumptions that underlie them should be an important part of any safety management system.

While it is unrealistic to expect people to recognize precursors that are only evident in hindsight, too often the precursors could and should have been identified and led to further investigation. *An effective SMS is characterized by continual vigilance against degradation of the safety control structure over time*.

A common paradox is that the fewer accidents an organizational has, the more likely it is that there will be more in the future. If the safety management system is effective, then losses are few. Unfortunately, people tend over time to judge the inherent amount of risk with respect to the current accident rate. If there are few accidents, namely, the safety management system is very effective, over time inherent risk is judged as low and safety management starts to degrade and become less effective. Feedback and information is required to keep the process models of decision makers up-to-date with the actual level of risk at any time.

Tips for risk management

- Create procedures for identifying hazards, analyzing them, and then using the resulting information in product development and operations. Assign responsibility to ensure this is taking place.
- Emphasize qualitative procedures that provide the information necessary for improvement and not just quantitative assessment. Don't make up quantitative assessments for likelihood that is unknowable.
- Reevaluate risk assessment results as data is obtained and when changes occur over time.
- Provide ways to identify when risk is increasing over what was originally assumed.
- When procedures are not being followed, don't just try to enforce them. Instead evaluate why they are not being followed and redesign them. Try to understand why gaps between what is specified and what is done are occurring.

Education and training

Everyone in the safety control structure, not just the lower-level controllers of the physical systems, must understand their roles and responsibilities with respect to safety and why the system—including the organizational aspects of the safety control structure—was designed the way it was. If employees understand the intent of the SMS and commit to it, they are more likely to comply with that intention rather than simply follow rules when it is convenient to do so. Training is not enough; education is required.

Education must include not only information about the hazards and safety constraints enforced by the controls, but also about priorities and how decisions are to be made. The safety philosophy statement, discussed earlier, provides information about the safety values to be used in decision making. In addition, everyone needs to know the risks they are taking in the decisions they make. Often, poor decision making arises from having an incorrect assessment of the risk being assumed. Using the non-probabilistic definition of risk provided above, this means that the decision makers must know how their decisions will impact the designed controls in the safety control structure.

Telling managers and employees to be "mindful of weak signals" (a common suggestion in the High Reliability Organization or HRO literature) simply creates a pretext for blame after a loss event occurs and hindsight provides the clarity that changes a weak signal (noise) into a strong signal. Instead, everyone must be trained on the hazards associated with the operation of a system and how to recognize them if we expect them to recognize the precursors to an accident. People need to know what to look for, not just be told to look for an undefined something.

Training should also include "why" as well as "what." Understanding the rationale behind the safety rules they are asked to follow will help reduce complacency, what appears to be reckless behavior (but to the person made perfect sense), and unintended changes leading to hazards. The rationale includes understanding why previous accidents occurred and what changes were made to try to prevent a reoccurrence. With increasing automation and people interacting with it, understanding the controlled hazards involved may require more training than was necessary in the past when the hazards were more obvious and intuitive. People who interact with complex systems need to learn more than just the procedures to follow: they must also have an in-depth understanding of the controlled physical process as well as the logic used in any automated controller (software) they may be supervising or with which they may be interacting. A list of what controllers—at all levels of the safety control structure—need to know was included in *Engineering a Safer World* and is repeated here:

- The system hazards and the reasons behind safety-critical procedures and operational rules.
- The potential result of removing or overriding controls, changing prescribed procedures; and inattention to safety-critical features and operations: Past accidents and their causes should be reviewed and understood.
- How to interpret feedback: Training needs to include different combinations of alerts and sequences of events, not just single events.
- How to think flexibly when solving problems: Controllers need to be provided with the opportunity to practice problem solving involving safety.
- General strategies rather than specific responses: Controllers need to develop skills for dealing with unanticipated events.
- How to test hypotheses in an appropriate way: To update mental models, human controllers often use hypothesis testing to understand the current system states and to update their process models. Such hypothesis testing is common with computers and automated systems where documentation is usually so poor and hard to use that experimentation is often the only way to understand the automation design and behavior. Hypothesis testing, however, can lead to losses. Designers need to provide operators with the ability to test hypotheses safely and controllers must be educated on how to do so.
- Emergency procedures must be overlearned and continually practiced. Controllers need to be taught about operating limits and specific actions to take in case they are exceeded. Requiring operators to make good decisions under stress and without full information simply ensures they will be blamed after a loss occurs.

Training should not be a one-time event for employees but should be continual throughout their employment, if only as a reminder of their responsibilities and the system hazards. Learning about recent events and trends should be part of this training. Assessing for the effectiveness of the training, perhaps through regular audits and performance assessments, can be useful in implementing an effective improvement and learning process. Incident and accident investigation results are an important source of information about training effectiveness.

Some companies have created a practice where managers provide safety training. Training experts help manage group dynamics and curriculum development, but the training is provided by project or group leaders. By learning to teach the materials, supervisors and managers are more likely to absorb and practice the key principles. In addition, it has been found that employees pay more attention to a message delivered by their boss than by a trainer or safety expert.

Tips for education and training

- Educate, don't just train
- Everyone should understand their roles and responsibilities, why the system was designed the way it was, information about hazards and safety constraints enforced by the controls, and the risks they are taking in the decisions they make.
- Training should include "why" and not just "what."
- Everyone should learn:
  - The system hazards and the reasons behind safety-critical procedures and operational rules that are related to their jobs.
  - The potential result of removing or overriding controls, changing prescribed procedures, and inattention to safety-critical features and operations.
- Make sure the causes of past accidents are well disseminated and understood.
- Give people the opportunity to practice problem solving involving safety
- Teach general strategies rather than specific responses so controllers can develop skills for dealing with unanticipated events.
- Teach operators how to test hypotheses in an appropriate way and provide ways for them to learn in this way.
- Emergency procedures should be overlearned and continually practiced.
- Contingency procedures should be reviewed before any safety-critical activity.
- Training should be continuous and not just a one-time event when hired.
- Teach about recent events and trends
- Consider having managers provide safety training.

Learning and continual improvement:

Because SMS designs are rarely perfect from the beginning and the world changes over time, there must be an effective process in place to ensure that the organization is continually learning from safety incidents and improving the SMS itself as well as the workplace, the products and the services.

Experimentation is an important part of the learning process, and trying new ideas and approaches to improving safety should be encouraged but also evaluated carefully to ensure that improvement actually results.

The Safety Information System (SIS)

A comprehensive and usable safety information system is key to having a successful SMS. It provides a source of information about the effectiveness of the control structure. In STAMP terms, this is the definition of risk.

The process models of controllers and decision makers must be kept accurate and coordinated. In essence, the SIS acts as a shared process model and a source for updating individual process models. Therefore, accurate and timely feedback and data are important. The SIS can provide the information necessary to detect trends, changes and other precursors to an accident; to evaluate the effectiveness of the safety controls; to compare models and risk assessments with actual behavior; and to learn from events and improve the SMS. After major accidents, it is often found that the information to prevent the loss existed but was not used or was not available to those involved. In general, lots of information is collected only because it is required for government reports and not necessarily because it is a necessity for the operation of an effective SMS.

To determine what should be in the SIS, the responsibilities of those in the safety control structure and the feedback they need to fulfill those responsibilities can be used, that is, the information required in their process/mental models in order to make appropriate decisions. In general, the SIS contains, at a minimum:

- The safety management plan (for both development and operations),
- The status of all safety-related activities for the system,
- The safety constraints and assumptions underlying the design, including operational limitations,
- The results of the hazard analyses (hazard logs) along with tracking and status information on all known hazards,
- The results of performance audits and assessments,
- Incident and accident investigation reports and corrective actions taken,
- Lessons learned and historical information, and
- Results of trend analysis.

Using the safety control structure design to identify what each person in it requires for safe decision making will help to design a usable SIS where each person is able to find the information they need.

Information may be collected by companies or by industries. No matter how the information is collected, understanding its limitations is important. To be most useful, the information must be accurate and timely and it must be disseminated to the appropriate people in a useful form. Three factors are involved: collection, analysis, and dissemination.

Collection: Data may be distorted by the way it is collected. Common problems are systematic filtering, suppression, and unreliability. Data collected for accident reports tend to focus on proximal events and actors but not the systemic factors involved such as management problems or organizational deficiencies. Studies[35] have shown that data from near-miss reporting by operators are filtered and primarily point to operator error as the cause of the events. Reports by management are similarly filtered and also point to operator error as the cause. Even general safety inspections tend to identify limited categories of conditions, especially when checklists are used.

Data collection tends to be more reliable for accidents that are similar to those that have occurred in the past than for new types of systems where past experience on hazards and causal factors is more limited. Software errors and computer problems are often omitted or inadequately described because of lack of knowledge, lack of accepted and consistent categorizations for such errors, or simply not considering them seriously as a causal factor.

Experience has shown that it is difficult to maintain reliable reporting of incidents over an extended period, and therefore this type of data is not very useful for estimating probabilities of events or their potential consequences. There is data to suggest that in some industries, up to three quarters of accidents may be unreported. Sometimes, information is involved that a company wants to keep secret or that those making the reports may worry will be used against them in job performance evaluations. Potential legal liability may also be involved.

Discussions about operational problems that have occurred should be documented in the SIS and analyzed for their hazard potential. While it is common, as previously discussed, for concerns to be brought directly to those who can address them, bypassing documentation in the SIS, very often these discussions or emails may contain safety issues that are not recognized as such by the participants. In addition, documenting these discussions can be beneficial as they contain valuable corporate knowledge that would otherwise be lost as people leave positions or retire or due to email retention standards. A

[35] Many scientific references for this section can be found in Nancy G. Leveson, *Safeware*, Addison Wesley Publishers, 1995, Chapter 11. We tried to keep this handbook from reading like an academic paper.

requirement to retain and document these communications should be implemented or a corporate communication channel that automatically retains such information should be developed.

Some measures can be used to improve the comprehensiveness and reliability of data collection such as special training of data collectors, feedback of results to the data collectors, fixed routines, and anonymous reporting. Training on techniques such as CAST can encourage the inclusion of information that is often omitted. Automated monitoring systems are now commonplace, but a problem is that they can and usually do collect so much information that it is difficult to analyze the results and detect problems. STPA might be a useful way to help determine what types of information need to be collected and to identify leading indicators in the large amounts of data collected.

Analysis: Systematizing and consolidating a large mass of data into a form useful for learning is difficult. Raw quantitative data can be misleading, especially if statistical significance is not included: Data is not the same as information. Again, STPA results can be used not only to identify what data needs to be collected, but to provide guidance on the importance of the events that are occurring. In addition, data to validate the STPA causal analysis results and to identify factors that were thought to be eliminated or mitigated should be part of the SIS collection and analysis process.

The biggest problem in analysis is simply that while it is easy to design data collection channels, finding the time and manpower to analyze all the data that results may be difficult or impractical. Tools like STPA can help to identify the most important data to collect and the most useful analysis processes.

Dissemination: Disseminating information (not data) in a useful form may be the most difficult part of an SIS. Information needs to be presented in a form that people can learn from, apply to their daily jobs, and use throughout the life cycle of projects, not just in the conceptual design stage. And it must be updated in a timely manner: accidents have resulted from changes during operations or due to insufficient updates to the hazard analysis when engineering modifications are made. STPA stresses usable documentation that includes rationale and intent as well as traceability to assist in the change process. Also, the assigned safety-related responsibilities to individuals and groups in the safety control structure will provide the information necessary to tailor information to the needs of those that receive it. Providing too much information to individuals can be as dangerous as providing too little.

The method for presenting information should be adaptable to the cognitive styles of the users and should be integrated into the environment in which safety-related decisions are made. Again, the safety control structure design process will provide a great deal of useful input about what information is needed, when it is needed, and how it will be used by individual controllers in the control structure.

Tips for designing a safety information system

- Accurate and timely feedback and data are important
- The SIS should provide the information necessary to detect trends, changes and other precursors to an accident; to evaluate the effectiveness of the safety controls; to compare models and risk assessments with actual behavior; and to learn from events and improve the SMS.
- Use the defined responsibilities of those in the safety control structure to identify the information they need to keep their process models accurate enough for good decision making.
- Understand the limitations of your collected data.
- To be most useful, information must be accurate and timely and it must be disseminated to the appropriate people in a useful form.
- Find ways to collect data that minimize distortion of the data (filtering, suppression, and unreliability).
- Keep detailed information on actual safety-related incidents and ensure that information gets to the people who need it.
- Create ways to improve the comprehension and reliability of data collection.
- Try to include statistical significance on numeric data if possible.
- Use STPA to identify what data needs to be collected and provide guidance on the importance of the events that are occurring.
- Collect data to validate the STPA causal analysis results and to identify factors that were thought to be eliminated or mitigated.
- Ensure that data is analyzed and not just collected.
- Present data in a way that people can learn from it and apply it to their daily jobs.
- Keep data up to date.
- Document design rationale and intent and provide traceability to assist in the change process.
- Tailor the information provided and the presentation format to the needs of those receiving it.
- Providing too much data, particularly raw data, can be as dangerous as providing too little.
- Adapt the presentation of information to the cognitive styles of the users and integrate it into the environment in which safety-related decisions are made.
- Use the safety control design process to determine what information is needed, when it is needed, and how it will be used.

Summary

In general, effective safety management requires:

- Commitment and leadership at all levels
- A strong corporate safety culture
- A clearly articulated safety vision, values and procedures, shared among stakeholders
- A safety control structure with appropriate assignment of responsibility, authority, and accountability.

- Feedback channels that provide an accurate view of the state of safety at all levels of the safety control structure
- Integration of safety into development and line operations (not just a separate and independent group or a separate subculture)
- Individuals with appropriate knowledge, skills, and ability
- Stakeholders with partnership roles and responsibilities
- A designated process for resolving tensions between safety priorities and other priorities
- Risk awareness and communication channels for disseminating safety information
- Controls on system migration toward higher risk
- An effective and usable safety information system
- Continual improvement and learning
- Education, training, and capability development

For more information, the reader is referred to Chapter 13 of *Engineering a Safer World* and to Chapters 11 and 12 in *Safeware*. Appendix D of this handbook also contains a list of the responsibilities that need to be included in an effective Safety Control Structure.

Chapter 8: Introducing STPA into Your Organization

John Thomas

Nancy Leveson

For relatively small organizations, introducing STPA may simply involve training and perhaps demonstration projects to establish that the practicality and advantages of using STPA are worthwhile. For larger organizations, other challenges may be involved such as changing the culture, changing the standard processes used, and training large numbers of people to do something new and different than they have in the past. This chapter examines some issues that arise when an organization decides to evaluate or adopt STPA for widespread use throughout the organization.

Training

Learning STPA requires some practice and a "learning by doing" approach. Formal, in-person training is very effective, and we've found that professionals can be trained and ready to begin trying STPA in just a few days. The most effective STPA training is interactive with hands-on exercises used to reinforce the process. Surprisingly, people who have been using traditional hazard analysis techniques may have the most difficulty in learning STPA—there seems to be a fair amount of unlearning required and extra training may be necessary. System engineers tend to learn the fastest in our experience.

There are tradeoffs that depend on the size of the training class. Small classes (under 10 people) can provide an in-depth and detailed understanding of STPA and can be a good option for small organizations that want to begin a small pilot study to evaluate STPA. Large classes (over 50 people) take longer and limit the amount of interaction that is possible, but can be a good option for organizations that need to provide a basic awareness of STPA to a large group. Medium classes (around 20-30 people) tend to offer a reasonable compromise.

For very large organizations in which hundreds or even thousands of people need to be trained, a "train the trainers" approach may be desirable. The most effective way to produce STPA experts who can serve as future trainers and facilitators (see below) is to immerse candidates in real projects where STPA is actively used. Attending a short training class is not enough to produce STPA experts, but those who have been immersed in a few large STPA projects are great candidates for future trainers and facilitators. One approach that has worked well is to allow one or more facilitators-in-training to shadow other STPA facilitators working on real projects. The facilitators in training learn a great deal by seeing first-hand the challenges encountered in different projects and the questions that are raised. Once they demonstrate the ability to provide effective guidance, solve challenges, and answer detailed STPA questions from other participants, they may be ready to facilitate on their own.

This handbook should provide more training options that were not possible before it was available. In addition, we are producing videos to go with the handbook.

Facilitators

STPA is most useful when integrated into the design effort and not performed as an independent effort by a separate group. However, not everyone will want or need to be trained as an STPA expert. The idea of a facilitator is used in HAZOP for the same reasons we encourage it for STPA, that is, experts on the system design make the best STPA team participants but may need guidance in actually applying the technique if they are not familiar with it.

The STPA facilitator's job is to provide STPA expertise and guidance throughout the process and to help avoid common traps that may be encountered. The facilitator can provide guidance during project selection and scoping; can help identify initial team members and the expertise that will be needed; may develop an initial set of losses, hazards, and potentially an initial control structure based on previous similar projects; and provide overall coordination throughout the process. For large projects, the facilitator will break up the process into smaller components that can be done successfully by individuals or by small groups of experts. The facilitator can identify and schedule tasks that may be done in parallel by different groups, and eventually the facilitator can bring these components together into the overall analysis. Facilitators may also lead the meetings of the participants and manage the interactions among the team members.

The facilitator may provide some initial STPA training at the start of the project, and they should be able to answer any STPA-related questions that arise during the project. The facilitator generally reviews the results during the process and at the end of the process to ensure the method is being followed correctly and that no gaps are overlooked. At the conclusion of the project, the facilitator may compile the results into a final report and ensure the findings are provided to the appropriate people.

Team Composition

STPA is best performed by a small team that includes different types of experts. An initial control structure can help identify the areas of expertise that will be needed. The interdisciplinary team typically includes expertise from different areas such as a software expert, an operations expert, a maintenance expert, etc. In some cases it may not be possible to include all the relevant experts on the STPA team (for example when the experts are in high demand). In that case you may want to ensure that the team will at least have access to people with the right technical knowledge and that you can get answers to questions that arise during the analysis.

Personalities matter. The engineers and designers of a system may have the most expertise, but they can sometimes be defensive and see the STPA effort as a challenge to the quality of their work. They may not want to contribute potential UCAs because they believe they have already been mitigated in the design, or they may argue that identified UCAs were assumed to be impossible and should be removed or not analyzed. On the other hand, outsiders or other designers not involved in this particular design may have less knowledge about the system but they may not be as defensive and may be happy to propose potential flaws. It may help to explain to engineers and designers that STPA is a worst-case analysis method and that it can help make their job easier by avoiding rework, not harder. It is also helpful to begin applying STPA as early as possible, before the design is completed and before decisions have already been made. It is most advantageous when STPA is applied by the project engineers themselves to help guide decisions and make informed choices. In any case, personalities matter and ideally you should try to find knowledgeable experts who are open to new approaches and who aren't hesitant to identify problems that may have been overlooked previously.

We have often found that initial skepticism by those deeply involved in a project design is quickly overcome when they find UCAs or scenarios that had not been previously identified. They then quickly become STPA promoters.

Those with background or experience in testing can be excellent choices for an STPA team. Testers can be quite knowledgeable about what can go wrong and usually they are not only willing but eager to find problems that may have been overlooked. Testers rarely assume that requirements, specifications, or design decisions are correct and their job involves constantly questioning the assumptions and claims that have been made. These are all excellent characteristics for STPA team members.

Cost and Return on Investment

STPA is very efficient. Figure 8.1 shows the relative amount of time spent by all STPA team members on a recent industry project. For most industry STPA projects, most of the time is actually not spent learning STPA or performing STPA but in learning how the system to be analyzed works (which would need to be done for any method that identifies hazard causes). About a quarter of the time is spent finding answers to "what if" technical questions about the system that had not have been considered previously. A relatively small amount of time is typically spent actually learning STPA and following the process.

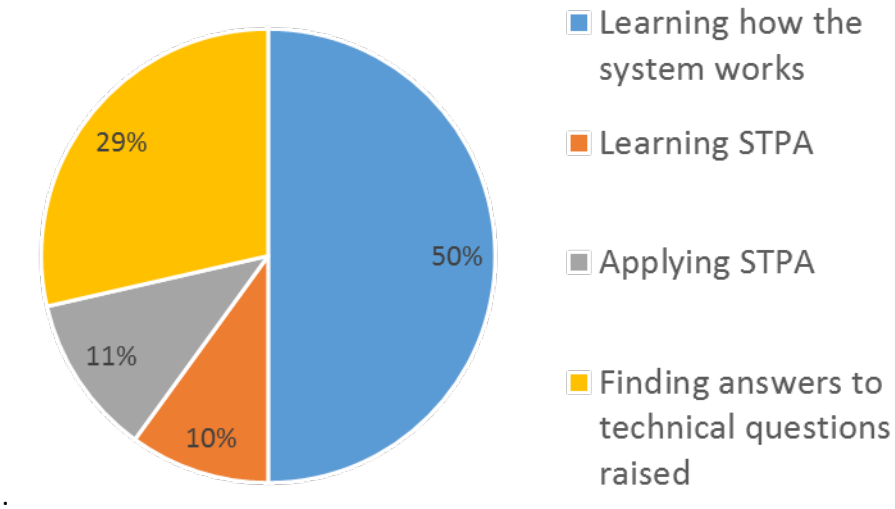


Figure 8.1: Relative amount of time spent on different tasks during a recent industry STPA project

STPA is not a paper-pushing exercise. Most of the time is spent exploring real concerns in the system that can immediately be used to drive decisions and provide insight about how to make improvements.

Although we do not have a lot of carefully collected data comparing the time required, people are reporting that STPA requires about 2 or 3 orders of magnitude less time and resources than the more traditional hazard analysis techniques and produces more complete results. Typically, the first use is most costly because of learning costs, but the cost quickly drops off on repeat projects.

An Example Introductory Use of STPA in a large organization

Chapter 4 describes the use of STPA in one large organization for a complex, highly automated system. We repeat it here. In this case, STPA was applied by an eight-member cross-functional team composed of members from integration, engineering, operations, and maintenance. All the people involved were considered to be leads or subject matter experts within their scope of responsibility for the system. The facilitator was an MIT master's student who was trained in STPA but had little knowledge about the new automated manufacturing process being introduced to the plant. The other members of the team had no knowledge of STPA at the beginning of the project.

The analysis was completed over the course of a three-week period. A group setting for the assessment was only used twice during the process for less than three hours each, while the rest of the assessment was completed in one-on-one meetings between the facilitator and the respective expert for the portion of the system being discussed.

Because STAMP and STPA were relatively new for the company and entirely new for the team, the first group meeting was used as an instruction period to provide an overview of the methodology being used and then to define the accidents (or losses) the team wanted to avoid, the hazards and system boundaries, and an initial draft of the safety control structure for this application.

The initial control structure was developed by the group as a whole to ensure that all of the major components were captured. After developing the first draft of the control structure, follow-on meetings were scheduled with individual experts or small groups with the appropriate technical expertise on the different parts of the system to add detail.

Once the appropriate level of control structure detail was developed, the one-on-one or small group discussions continued with the identification of Unsafe Control Actions and scenarios. The results of this portion of the analysis were then discussed in another group setting to confirm the individual findings of the group and discuss potential mitigations for the system. This process occurred over the course of three weeks, and the time spent in group, one-on-one meetings, and individual facilitator work totaled approximately 300 person hours.

In general, the scenarios identified included such causal factors as hardware failures, incorrect process models (stemming from missing or incorrect feedback), system component interactions, and the role of management and procedures on safety combined with schedule pressures in the factory. After the team identified STPA scenarios, they used the results to develop new controls for the system in order to eliminate or prevent entering hazardous states. Examples are included in Chapter 4 of this handbook.

Because STPA uses a model of the control structure, the assessment can focus on specific control loops within the model after the complete model is constructed. This ability to divide up the modeling and analysis process provided for easy scheduling of meeting times and numerous meetings within the available time frames of everyone within the group. During these meetings, details such as control actions, process models, feedback, etc. were added and any conflicts or issues were resolved with the affected parties.

Other Advice

We intend to include more information about integrating STPA into large organizations as we obtain it, including return on investment data. The fact that so many organizations are adopting STPA after trying it on several projects and comparing it with what they do now is encouraging. If you have any input to provide, please tell us.

Appendix A : Examples of Hazards

Hazards are often very similar within an industry. Once you have identified the hazards appropriate for your industry, product, or services, you are likely to be able to reuse the list with perhaps small changes. This appendix contains some example lists of hazards that have been used on real projects. Of course, the stakeholders are the ultimate decision makers about what types of losses and hazards will be considered. The losses that are considered important to the stakeholders for a particular project will obviously affect the hazards identified.

Nuclear Power Plant

Losses:

L1: People injured or killed

L2: Environment contaminated

L3: Equipment damage (economic loss)

L4: Loss of electrical power generation

Hazards

H1: Release of radioactive materials [L1, L2, L3, L4]

H2: Reactor temperature too high [L1, L2, L3, L4]

H3: Equipment operated beyond limits [L3, L4]

H4: Reactor shut down [L4]

Aircraft

Losses:

L1. Loss of life or serious injury to people

L2. Damage to the aircraft or objects outside the aircraft

Hazards

H-1: Aircraft violate minimum separation standards in flight [L1, L2]

H-2: Controlled flight of aircraft into terrain [L1, L2]

H-3: Loss of aircraft control [L1, L2]

H-4: Aircraft airframe integrity is lost [L1, L2]

H-5: Aircraft environment is harmful to human health [L1, L2]

H-6: Aircraft departs designated taxiway, runway, or apron on ground [L1, L2]

H-7: Aircraft comes too close to other objects on the ground [L1, L2]

Radiation Therapy

Losses:

L1: The patient is injured or killed from overexposure or undertreatment.

L2: A nonpatient is injured or killed by radiation.

L3: Damage or loss of equipment.

L4: Physical injury to a patient or nonpatient during treatment.

Hazards:

H1: Wrong dose: Dose delivered to patient is wrong in either amount, location, or timing [L1].

H1a: Right patient, right dose, wrong location.

H1b: Right patient, wrong dose, right location.

H1c: Right patient, wrong dose, wrong location.

H1d: Wrong patient.

H2: A nonpatient is unnecessarily exposed to radiation [L2]

H3: Equipment is subject to unnecessary stress [L3].

H4: Persons are subjected to nonradiological injury [L4].

Military Aviation

Mishaps

M-1: Loss of or damage to the aircraft or equipment on the aircraft

M-2: Serious injury or fatality to personnel

M-3: Inability to complete the mission

Hazards

H-1: Violation of minimum separation standards from fixed or moving objects [M-1, M-2, M-3]

H-2: Inability to control the aircraft [M-1, M-2, M-3]

H-3: Loss of airframe integrity [M-1, M-2, M-3]

Additional hazards may be relevant depending on the missions of the aircraft. As an example, for the weapon system on the aircraft:

H-4: Uncommanded detonation [M-1, M-2, M-3]

H-5: Uncommanded launch [M-1, M-2, M-3]

H-6: Collateral damage or friendly fire [M-1, M-2, M-3]

H-7: Non-deployment (detonation and/or launch) of ordinance when commanded [M-3]

Automotive

Losses

L1: Loss of life or serious injury to people

L2: Damage to the vehicle or objects outside the vehicle

Hazards

H1: Vehicle does not maintain safe distance from nearby objects [L1, L2]

H2: Vehicle enters dangerous area/region [L1, L2]

H3: Vehicle exceeds safe operating envelope for environment (speed, lateral/longitudinal forces) [L1, L2]

H4: Vehicle occupants exposed to harmful effects and/or health hazards [L1, L2]
(e.g. fire, excessive temperature, inability to escape, door closes on passengers, etc.)

Appendix B: Example Hierarchical Control Structures

This appendix shows some control structures we have used on our own projects.

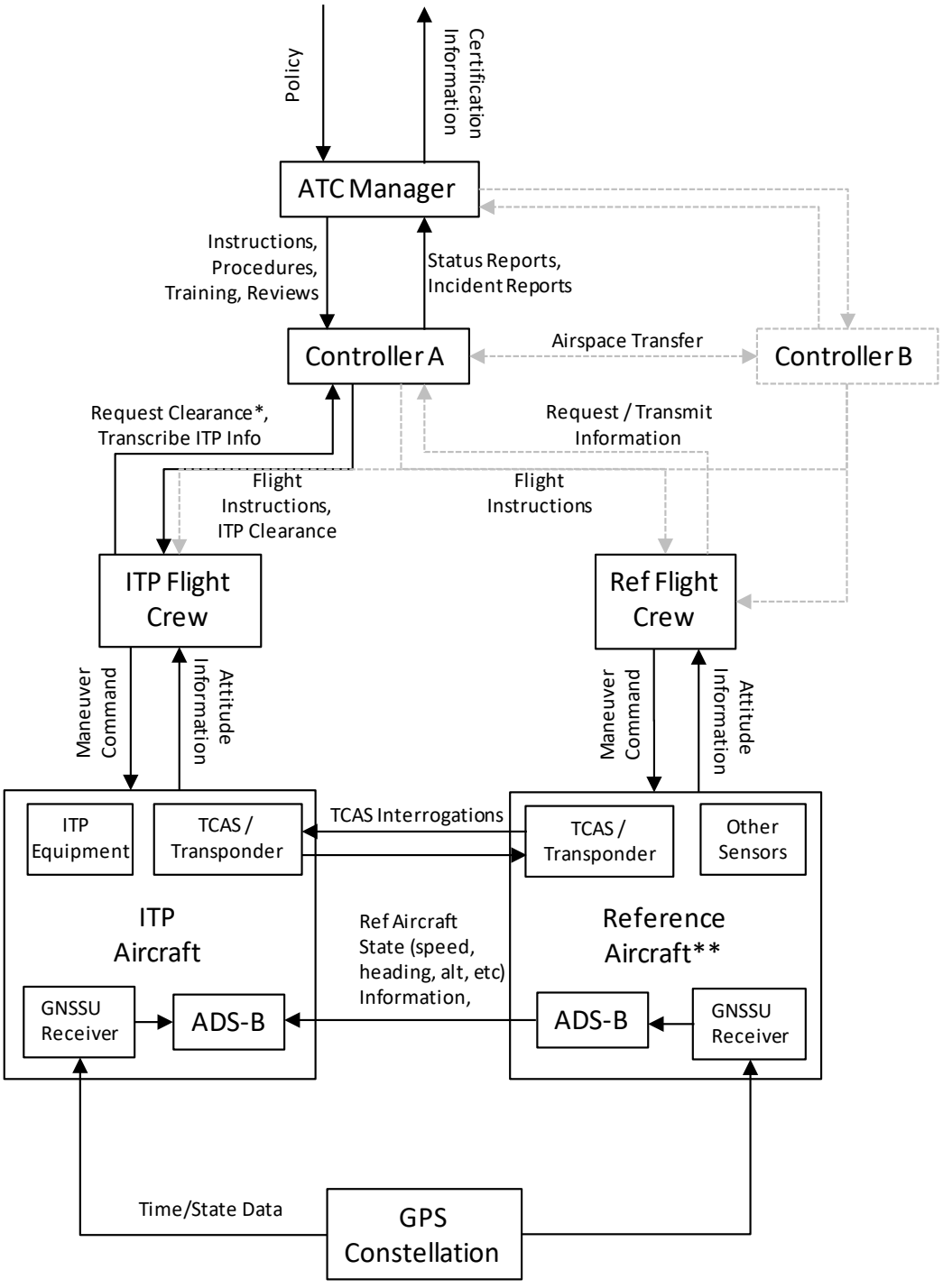


Figure B.1: Control structure for NextGen In-Trail Procedure with new equipment (aviation)

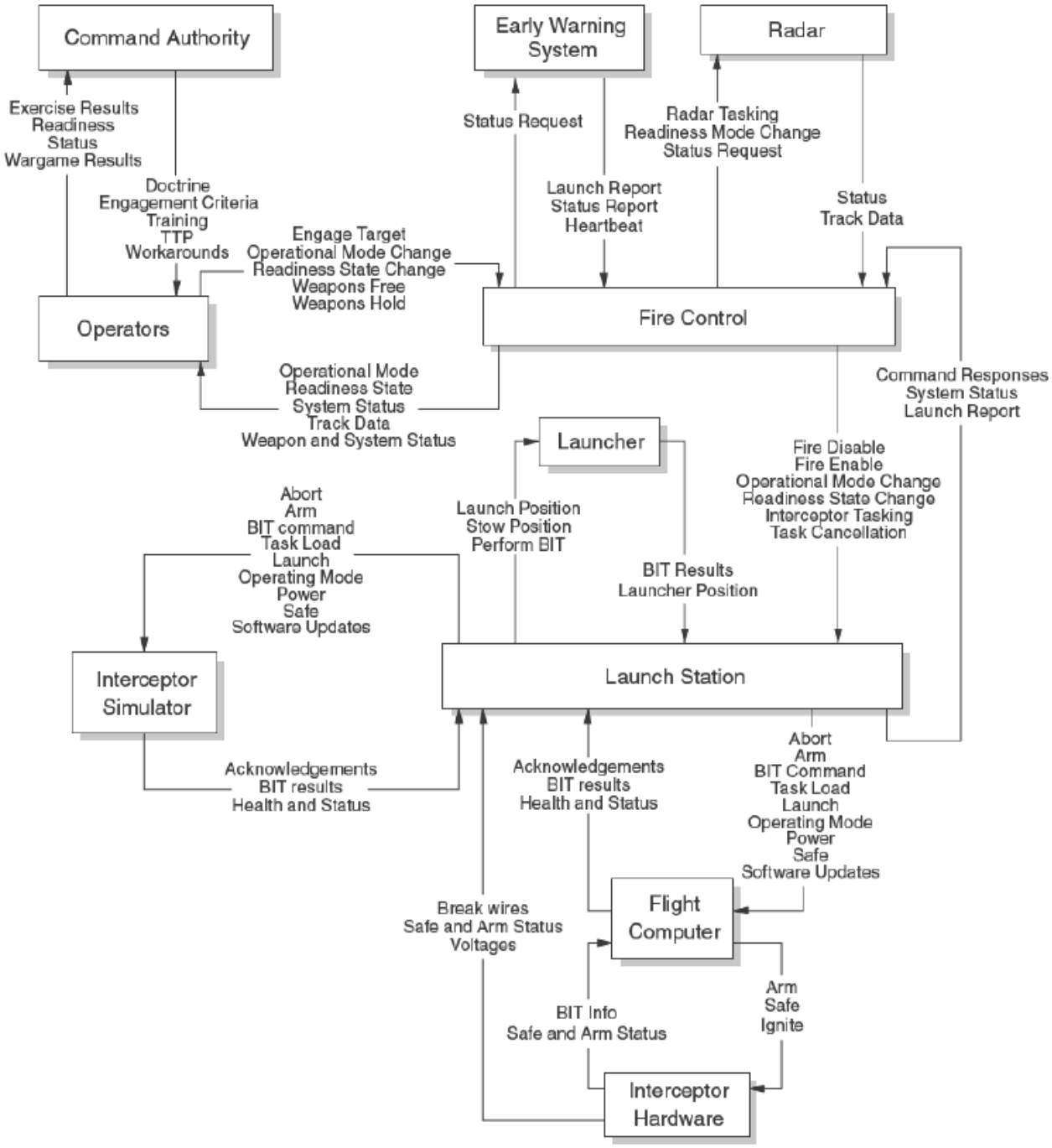


Figure B.2: Control structure for a Fictional Missile Intercept System

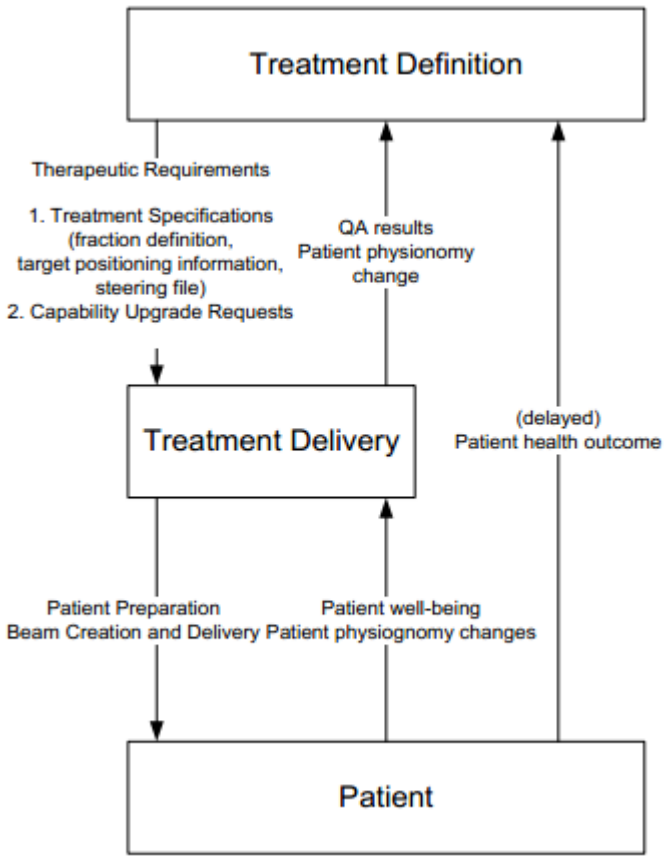


Figure B.3: High-level (abstract) control structure for a proton therapy machine

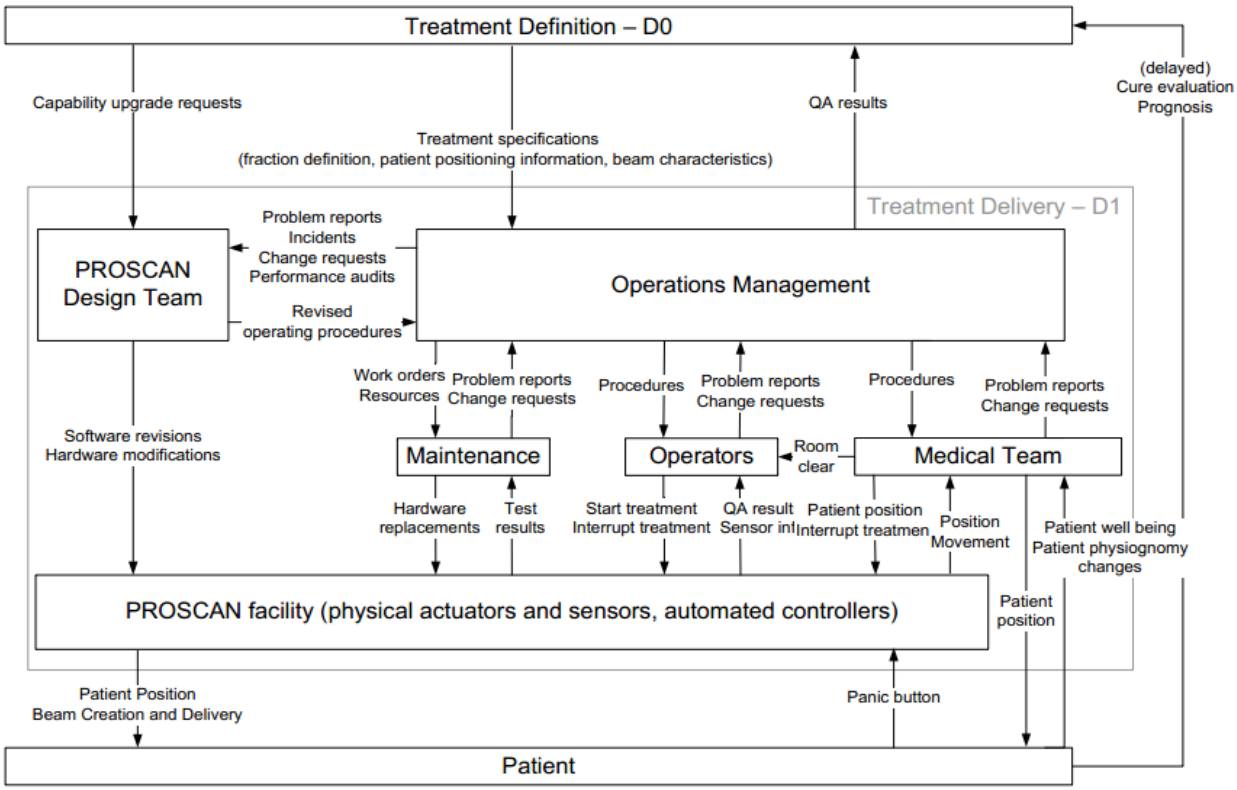


Figure B.4: Zooming into the Treatment Delivery part of Figure B.3 with the Treatment Definition and Patient components shown to provide context

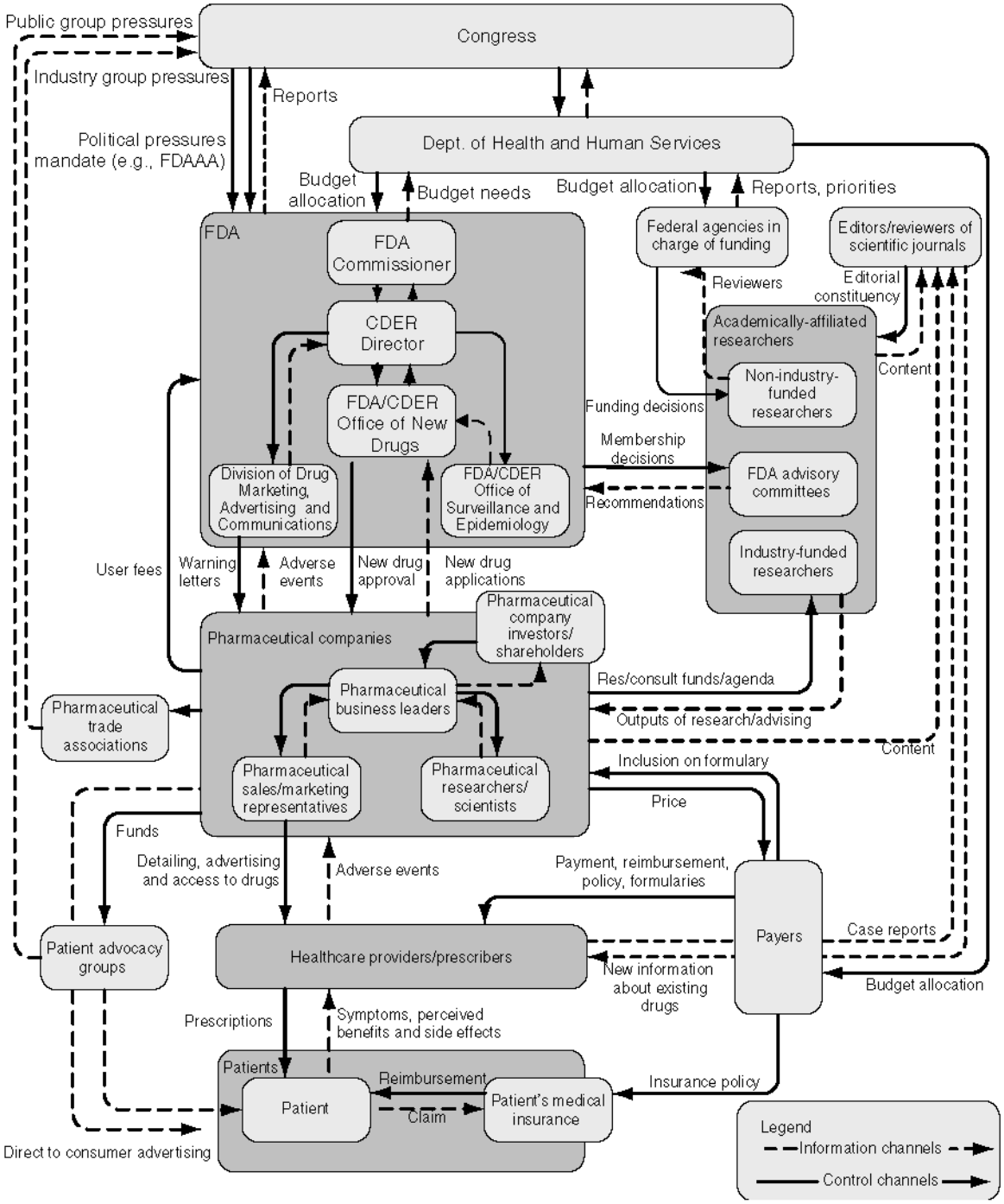


Figure B.5: U.S. Pharmaceutical Approval Safety Control Structure

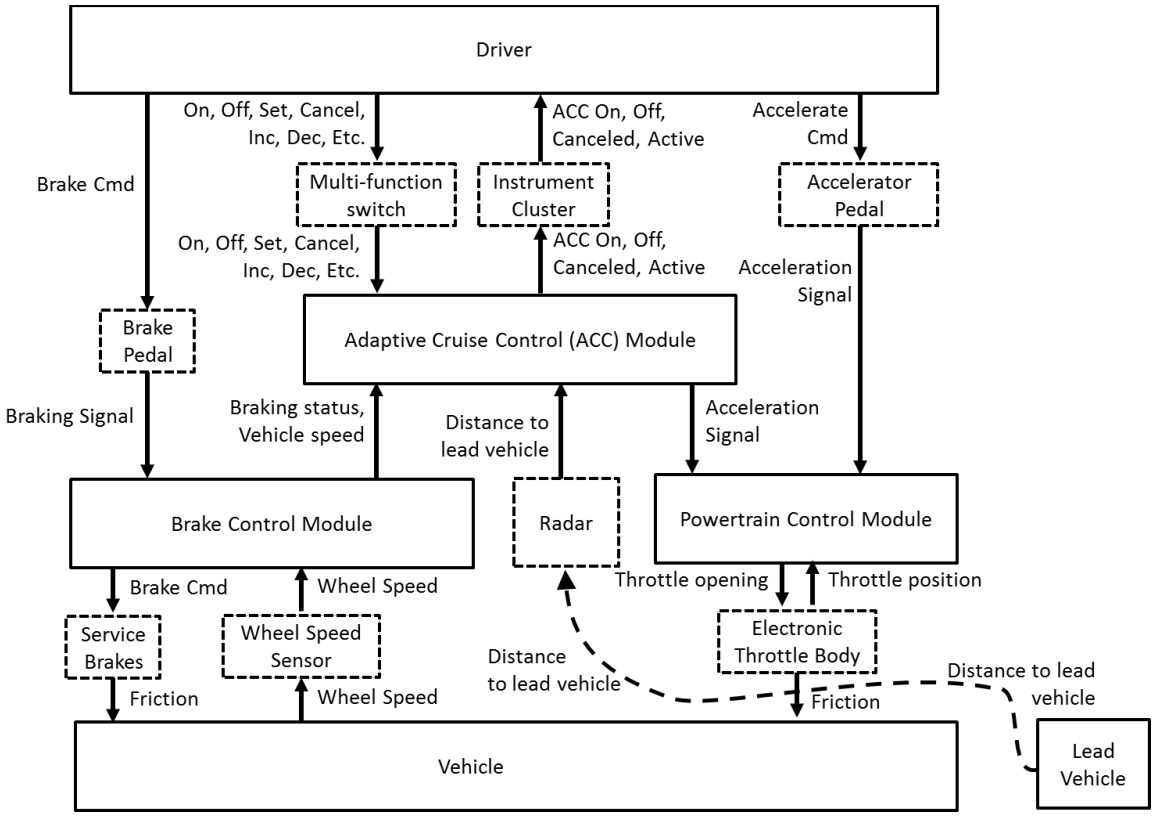


Figure B.6: Automotive Adaptive Cruise Control System

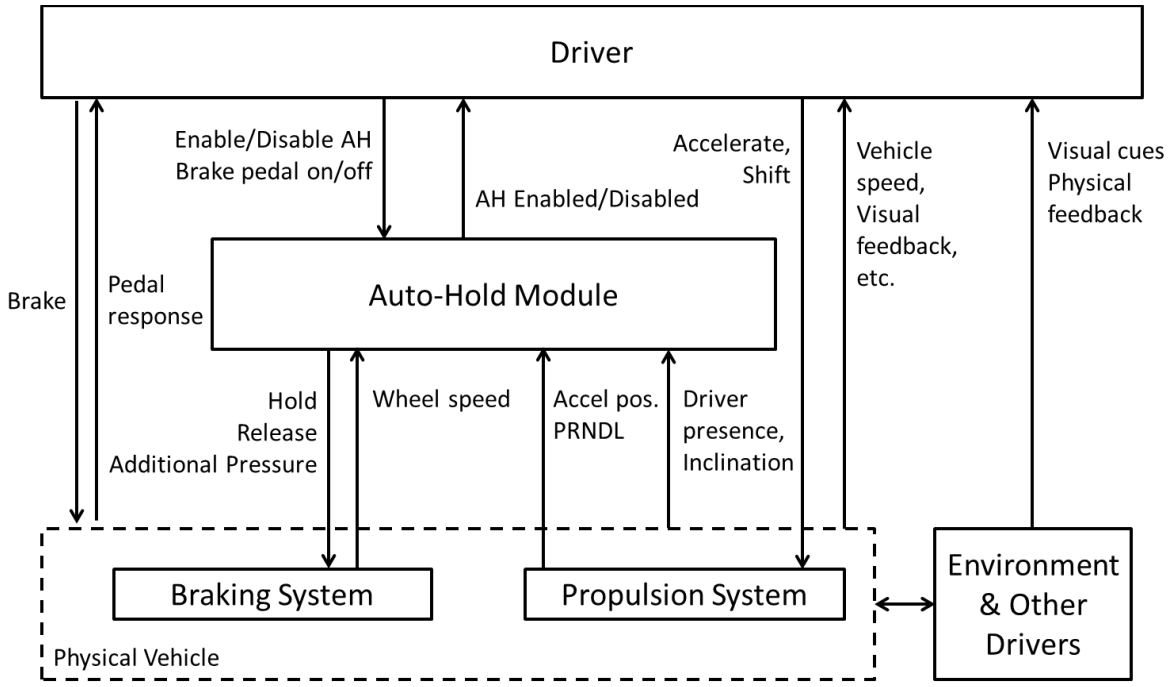


Figure B.7: Control Structure for Automotive Auto-Hold System

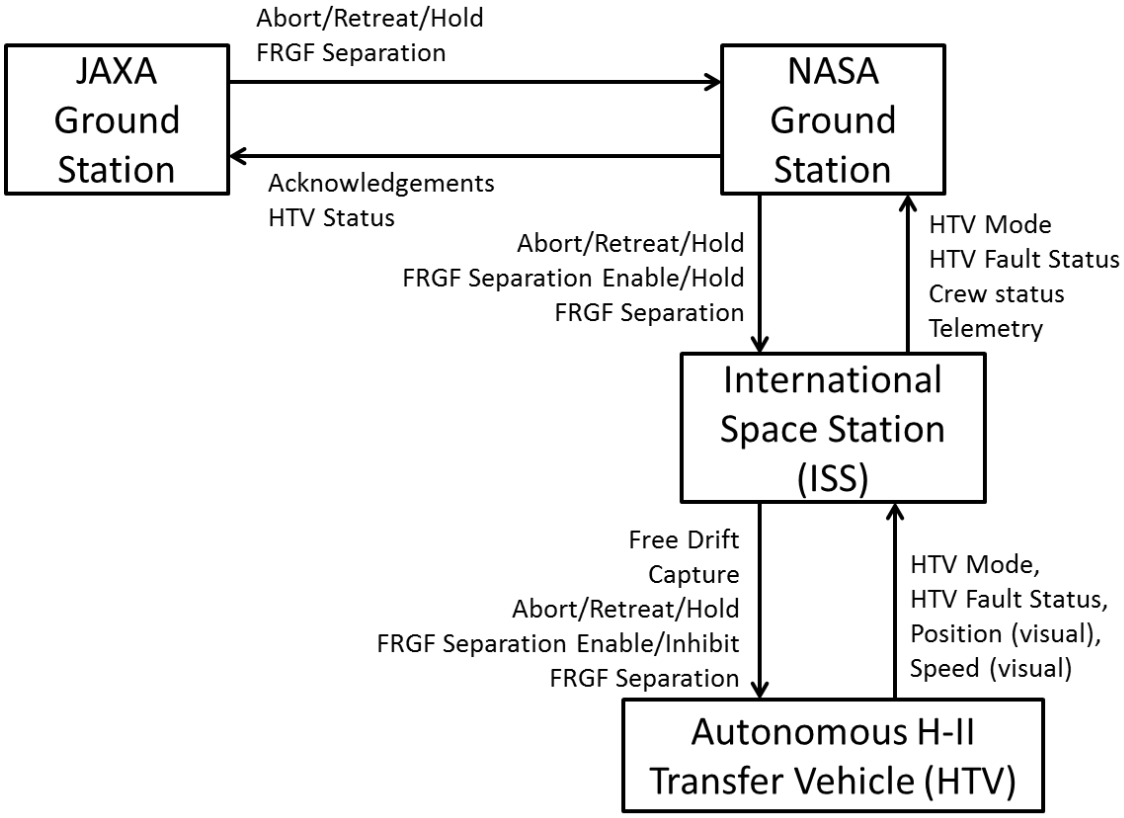


Figure B.8: Control structure for example autonomous space vehicle operations

Appendix C: More Examples of UCA Tables

Figure C.1: UCAs for Automotive Auto-Hold System

Control action (by Auto-Hold)	Not providing causes hazard	Providing causes hazard	Incorrect Timing/Order	Stopped Too Soon / Applied too long
Hold Command	UCA-AH-1: AH does not provide HOLD when vehicle stops and brake pedal released [H-1,2]	UCA-AH-2: AH provides HOLD when driver is applying the accelerator [H-1] UCA-AH-3: AH provides HOLD when AH is DISABLED [H-1] UCA-AH-4: AH provides HOLD when vehicle is moving [H-1] UCA-AH-5: AH provides HOLD when driver is not applying brake [H-1,2]	UCA-AH-6: AH provides HOLD too early before the required time at rest has not been met [H-1] UCA-AH-7: AH provides HOLD too late after vehicle stops, begins to roll [H-1]	N/A
Release Command	UCA-AH-10: AH does not provide RELEASE when driver applies accelerator pedal [H-1,2]	UCA-AH-12: AH provides RELEASE when driver is not applying accelerator [H-1,2]	UCA-AH-13: AH provides RELEASE too early before there is sufficient wheel torque [H-1,2] UCA-AH-13: AH provides RELEASE too late after accelerator applied, engine torque exceeds wheel torque [H-1,2]	N/A
Additional Pressure Command	UCA-AH-14: AH does not provide ADDITIONALPRESSURE when vehicle is slipping and AH is active [H-1,2]	UCA-AH-15: AH provides ADDITIONALPRESSURE when AH is not active [H-1,2] UCA-AH-16: AH provides ADDITIONALPRESSURE when brake system specs are exceeded [H-1,2]	UCA-AH-16: AH provides ADDITIONAL-PRESSURE too late after vehicle is slipping [H-1,2]	N/A

Table C.2: UCAs for autonomous H-II Transfer Vehicle (HTV) operations

Control Action (from ISS Crew)	Not providing causes hazard	Providing causes hazard	Too Early, Too Late, Order	Stopped Too Soon / Applied too long
Abort	ISS crew does not provide Abort Cmd when emergency condition exists [H-1]	ISS crew provides Abort Cmd when HTV is captured [H-1] ISS crew provides Abort Cmd when ISS is in Abort path [H-1]	ISS crew provides Abort Cmd too late to avoid collision [H-1] ISS crew provides Abort Cmd too early before capture is released [H-1]	N/A
Free Drift	ISS crew does not provide Free Drift Cmd when HTV is stopped in capture box [H-1]	ISS crew provides Free Drift Cmd when HTV is approaching ISS [H-1]	ISS crew provides Free Drift Cmd too late, more than X minutes after HTV stops [H-1] ISS crew provides Free Drift Cmd too early before HTV stops [H-1]	N/A
Capture	ISS crew does not perform Capture when HTV is in capture box in free drift [H-1]	ISS crew performs Capture when HTV is not in free drift [H-1] ISS crew performs Capture when HTV is aborting [H-1] ISS crew performs Capture with excessive/insufficient movement (can impact HTV, cause collision course) [H-1]	ISS crew performs Capture too late, more than X minutes after HTV deactivated [H-1] ISS crew performs Capture too early before HTV deactivated [H-1]	ISS crew continues performing Capture too long after emergency condition exists [H-1]

Table C.3: UCAs for Aircraft Flight Crew related to the Wheel Braking System

Control Action (by Flight Crew)	Not providing causes hazard	Providing causes hazard	Too soon, too late, out of order	Stopped too soon, applied too long
CREW.1 **Manual braking via brake pedals**	CREW.1a1: Crew does not provide manual braking during landing, RTO, or taxiing when Autobrake is not providing braking or is providing insufficient braking [H4.1]	CREW.1b1: Crew provides manual braking with insufficient pedal pressure [H4.1] CREW.1b2: Crew provides manual braking with excessive pedal pressure (resulting in loss of control, passenger/crew injury, brake overheating, brake fade or tire burst during landing) [H4-1, H4-5] CREW.1b3: Crew provides manual braking provided during normal takeoff [H4-2, H4-5]	CREW.1c1: Crew provides manual braking before touchdown (causes wheel lockup, loss of control, tire burst) [H4.1] CREW.1.c2: Crew provides manual braking too late (TBD) to avoid collision or conflict with another object (can overload braking capability given aircraft weight, speed, distance to object (conflict), and tarmac conditions)[H4-1, H4-5]	CREW.1d1: Crew stops providing manual braking command before safe taxi speed (TBD) is reached [H4.1, H4.4] CREW.1d2: Crew provides manual braking too long (resulting in stopped aircraft on runway or active taxiway) [H4-1]
CREW.2 **Arm autobrake**	CREW.2a1: Crew does not arm Autobrake before landing (causing loss of automatic brake operation when spoilers deploy. Crew reaction time may lead to overshoot.) [H4-1, H4-5]	CREW.2b1: Crew does not arm Autobrake to maximum level during takeoff. (assumes that maximum braking force is necessary for rejected takeoff) [H4-2]	CREW.2c1: Crew provides arm command too late (TBD) (resulting in insufficient time for BSCU to apply brakes) [H4-1, H4-5]	

	Crew.2a2: Crew does not arm Autobrake prior to takeoff (resulting in insufficient braking during rejected takeoff, assuming that Autobrake is responsible for braking during RTO after crew throttle down) [H4-2]	CREW.2b2: Crew armed autobrake with too high of a deceleration rate for runway conditions (resulting in loss of control and passenger or crew injury). [H4-1, H4-5]		
CREW.3 Disarm Autobrake	CREW.3a1: Crew does not disarm Autobrake during TOGA (resulting in loss of acceleration during (re)takeoff) [H4-1, H4-2, H4-5]	CREW.3b1: Crew disarms Autobrake during landing or RTO (resulting in loss of automatic brake operation when spoilers deploy. Crew reaction time may lead to overshoot) [H4-1, H4-5]	CREW.3c1: Crew disarms Autobrake more than TBD seconds after (a) aircraft descent exceeds TBD fps, (b) visibility is less than TBD ft, (c) etc…, (resulting in either loss of control of aircraft or loss of acceleration during (re)takeoff) [H4-1, H4-2, H4-5]	
CREW.4 Power off BSCU	CREW.4a1: Crew does not power off BSCU in the event of abnormal WBS behavior (needed to enable alternate braking mode) [H4-1, H4-2, H4-5]	CREW.4b1: Crew powers off BSCU when Autobraking is needed and WBS functioning normally [H4-1, H4-5] CREW.4b2: Crew powers off BSCU when Autobrake is needed (or about to be used) and WBS if funtioning normally [H4-1, H4-5]	CREW.4c1: Crew powers off BSCU too late (TBD) to enable alternate braking mode in the event of abnormal WBS behavior [H4-1, H4-5] CREW.4c2: Crew powers off BSCU too early before Autobrake or Anti-Skid behavior is completed when it is needed [H4-1, H4-5]	N/A

		CREW.4b3: Crew powers off BSCU when Anti-Skid functionality is needed (or will be needed) and WBS is functioning normally [H4-1, H4-5]		
CREW.5 Power on BSCU	CREW.5a1: Crew does not power on BSCU when Normal braking mode, Autobrake, or Anti-Skid is to be used and WBS functioning normally [H4-1, H4-5]		CREW.5c1: Crew powers on BSCU too late after Normal braking mode, Autobrake, or Anti-Skid is needed [H4-1, H4-5]	N/A

Table C.4: UCAs for Aircraft Brake System Control Unit (BSCU) Autobrake

Control Action (by BSCU)	Not providing causes hazard	Providing causes hazard	Too soon, too late, out of order	Stopped too soon, applied too long
BSCU.1 Autobrake Brake command	BSCU.1a1: Autobrake does not provide Brake command during RTO to V1 (resulting in inability to stop within available runway length) [H4-1, H4-5]	BSCU.1b1: Autobrake provides excessive Braking commands during landing roll [H4-1, H4-5]	BSCU.1c1 Autobrake provides Braking command before touchdown (resulting in tire burst, loss of control, injury, other damage) [H4-1, H4-5]	BSCU.1d1 Autobrake stops providing Brake during landing roll before TBD taxi speed attained (causing reduced deceleration)[H4-1, H4-5]
	BSCU.1a2 Autobrake does not provides Brake command during landing roll when BSCU is armed (resulting in insufficient deceleration and potential overshoot) [H4-1, H4-5]	BSCU.1b2 Autobrake provides Braking command inappropriately during takeoff (resulting in inadequate acceleration) [H4-1, H4-2, H4-5]	BSCU.1c2 Autobrake provides Brake command more than TBD seconds after touchdown (resulting in insufficient deceleration and potential loss of control, overshoot) [H4-1, H4-5]	BSCU.1d2 Autobrake provides Brake command too long (more than TBD seconds) during landing roll (causing stop on runway) [H4-1]
		BSCU1b3 Autobrake provides Brake command with insufficient level (resulting in insufficient deceleration during landing roll) [H4-1, H4-5]	BSCU1c3 Autobrake provides Brake command at any time before wheels have left ground and RTO has not been requested (brake might be applied to stop wheels before gear retraction) [H4-1, H4-2, H4-5]	BSCU.1d3 Autobrake provides Brake command for tire lock until less than TBD seconds before touchdown during approach (resulting in loss of control, equipment damage) [H4-1, H4-5]

BCSU.1a4

Autobrake does not provide Brake command after takeoff (needed to lock wheels, results in potential equipment damage during landing gear retraction or wheel rotation in flight) [H4-6]

BSCU.1c4

Autobrake provides Brake more than TBD seconds after V1 during rejected takeoff (assumes that Autobrake is responsible for braking during RTO after crew throttle down) [H4-2]

Appendix D: Responsibilities to be Included in the Safety Control Structure

The list below (from *Engineering a Safer World*, Chapter 13) can be used in creating a new safety control structure, as a checklist for those creating a model of their existing safety control structure or initiating an activity to improve it, and in identifying inadequate controls and control structures when performing incident and accident analysis. It is not meant to be exhaustive and will need to be supplemented for specific industries and safety programs.

This list only contains general responsibilities and does not indicate how they should be assigned. Appropriate assignment of the responsibilities to specific people and places in the organization will depend on the management structure of each organization. Each general responsibility may be separated into multiple individual responsibilities and assigned throughout the safety control structure, with one group actually implementing the responsibilities and others above them supervising, leading or directing, or overseeing the activity. Of course, each responsibility assumes the need for associated authority and accountability, as well as the controls, feedback, and communication channels necessary to implement the responsibility.

General Management

- Provide leadership, oversight, and management of safety at all levels of the organization.
- Create a corporate or organizational safety policy. Establish criteria for evaluating safety-critical decisions and implementing safety controls. Establish distribution channels for the policy. Establish feedback channels to determine whether employees understand it, are following it, and whether it is effective. Update the policy as needed.
- Establish corporate or organizational safety standards and then implement, update, and enforce them. Set minimum requirements for safety engineering in development and operations and oversee the implementation of those requirements, including any contractor activities. Set minimum physical and operational standards for hazardous operations.
- Establish incident and accident investigation standards and ensure recommendations are implemented and effective. Use feedback to improve the standards.
- Establish management of change requirements for evaluating all changes for their impact on safety, including changes in the safety control structure. Audit the safety control structure for unplanned changes and migration toward states of higher risk.
- Create and monitor the organizational safety control structure. Assign responsibility, authority, and accountability for safety.
- Establish working groups.
- Establish robust and reliable communication channels to ensure accurate management risk awareness of the development system design and the state of the operating process. These channels should include contractor activities.
- Provide physical and personnel resources for safety-related activities. Ensure that those performing safety-critical activities have the appropriate skills, knowledge, and physical resources.
- Create an easy-to-use problem reporting system and then monitor it for needed changes and improvements.
- Establish safety education and training for all employees and establish feedback channels to determine whether it is effective along with processes for continual improvement. The education should include reminders of past accidents and causes and input from lessons learned and trouble

reports. Assessment of effectiveness may include information obtained from knowledge assessments during audits.
- Establish organizational and management structures to ensure that safety-related technical decision making is independent from programmatic considerations, including cost and schedule.
- Establish defined, transparent, and explicit resolution procedures for conflicts between safety-related technical decisions and programmatic considerations. Ensure that the conflict resolution procedures are being used and are effective.
- Ensure that managers who are making safety-related decisions are fully informed and skilled. Establish mechanisms to allow and encourage all employees (including front-line operators) and contractors to contribute to safety-related decision making.
- Establish an assessment and improvement process for safety-related decision making.
- Create and update the organizational safety information system.
- Create and update safety management plans.
- Establish communication channels, resolution processes, and adjudication procedures for employees and contractors to surface complaints and concerns about the safety of the system or parts of the safety control structure that are not functioning appropriately. Evaluate the need for anonymity in reporting concerns.

Development

- Implement special training for developers and development managers in safety-guided design and other necessary skills. Update this training as events occur and more is learned from experience. Create feedback, assessment, and improvement processes for the training.
- Create and maintain the hazard log. Establish and maintain documentation and tracking of hazards and their status.
- Establish working groups.
- Design safety into the system using system hazards and safety constraints. Iterate and refine the design and the safety constraints as the design process proceeds. Ensure the system design includes consideration of how to eliminate or reduce contextual factors that cause or contribute to unsafe operator behavior that, in turn, contributes to system hazards. Distraction, fatigue, etc. are risk factors resulting from design that is dependent on humans performing in a way the designer imagined they would rather than behaving as normal humans would in such situations.
- Document operational assumptions, safety constraints, safety-related design features, operating assumptions, safety-related operational limitations, training and operating instructions, audits and performance assessment requirements, operational procedures, and safety verification and analysis results. Document both what and why, including tracing between safety constraints and the design features to enforce them.
- Perform high-quality and comprehensive hazard analyses to be available and usable when safety-related decisions need to be made, starting with early decision making and continuing through the system's life. Ensure that the hazard analysis results are communicated in a timely manner to those who need them. Establish a communication structure that allows communication downward, upward, and sideways (i.e., among those building subsystems). Ensure that hazard analyses are updated as the design evolves and test experience is acquired.
- Train engineers and managers to use the results of hazard analyses in their decision making.
- Maintain and use hazard logs and hazard analyses as experience is acquired. Ensure communication of safety-related requirements and constraints to everyone involved in development.

- Gather lessons learned in operations (including accident and incident reports) and use them to improve the development processes. Use operating experience to identify flaws in the development safety controls and implement improvements.

Operations

- Create an operations safety management plan
- Develop special training for operators and operations management to create needed skills and update this training as events occur and more is learned from experience. Create feedback, assessment, and improvement processes for this training. Train employees to perform their jobs safely, understand proper use of safety equipment, and respond appropriately in an emergency.
- Establish working groups.
- Maintain and use hazard logs and hazard analyses during operations as experience is acquired.
- Ensure all emergency equipment and safety devices are operable at all times during hazardous operations. Before safety-critical, non-routine, potentially hazardous operations are started, inspect all safety equipment to ensure it is operational, including the testing of alarms.
- Perform an in-depth investigation of any operational anomalies, including hazardous conditions (such as water in a tank that will contain chemicals that react to water) or events. Determine why they occurred before any potentially dangerous operations are started or restarted. Provide the training necessary to do this type of investigation and proper feedback channels to management.
- Create management of change procedures and ensure they are being followed. These procedures should include hazard analyses on all proposed changes and approval of all changes related to safety-critical operations. Create and enforce policies about disabling safety-critical equipment.
- Perform safety audits, performance assessments, and inspections using the hazard analysis results as the preconditions for operations and maintenance. Collect data to ensure safety policies and procedures are being followed and that education and training about safety is effective. Establish feedback channels for leading indicators of increasing risk.
- Use the hazard analysis and documentation created during development and passed to operations to identify leading indicators of migration toward states of higher risk. Establish feedback channels to detect the leading indicators and respond appropriately.
- Establish communication channels from operations to development to pass back information about operational experience.
- Perform in-depth incident and accident investigations, including all systemic factors. Assign responsibility for implementing all recommendations. Follow up to determine whether recommendations were fully implemented and effective.
- Perform independent checks of safety-critical activities to ensure they have been done properly.
- Prioritize maintenance for identified safety-critical items. Enforce maintenance schedules.
- Create and enforce policies about disabling safety-critical equipment and making changes to the physical system.
- Create and execute special procedures for the startup of operations in a previously shutdown unit or after maintenance activities.
- Investigate and reduce the frequency of spurious alarms.
- Clearly mark malfunctioning alarms and gauges. In general, establish procedures for communicating information about all current malfunctioning equipment to operators and ensure they are being followed. Eliminate all barriers to reporting malfunctioning equipment.
- Define and communicate safe operating limits for all safety-critical equipment and alarm procedures. Ensure that operators are aware of these limits. Assure that operators are rewarded for

following the limits and emergency procedures, even when it turns out no emergency existed. Provide for tuning the operating limits and alarm procedures over time as required.

- Ensure that spare safety-critical items are in stock or can be acquired quickly.
- Establish communication channels to plant management about all events and activities that are safety-related. Ensure management has the information and risk awareness they need to make safe decisions about operations.
- Ensure emergency equipment and response is available and operable to treat injured workers.
- Establish communication channels to the community to provide information about hazards and necessary contingency actions and emergency response requirements.

Appendix E: Limitations in Analytic Decomposition for Software-Intensive Systems (an Avionics Example)

Classic approaches to ensuring a property in a system apply analytic decomposition, that is, the individual components are analyzed and the results are combined to ensure the property exists in the system as a whole. This approach works for systems where the coupling and interactions among components are relatively simple, an assumption that primarily holds for electro-mechanical systems.[36] For software-intensive systems, the assumption underlying this approach about independence and reducibility (e.g., the components do not have indirect or unintended impact on each other) do not hold. The aircraft industry is starting to discover this fact. A paper by Bartley and Lingberg[37] describes the problem and provides examples of occurrences on aircraft in flight. They discuss the problem in the context of Integrated Modular Avionics (IMA) systems, which is a set of aircraft avionics functions that may be created by different manufacturers and updated individually over time.

To allow shared information in an IMA system, interfaces are typically specified in an Interface Control Document (ICD) for electrical interactions, mechanical or physical interactions, and software module interactions. A typical software interface specification includes things such as digital signal characteristics; data transmission format, coding, timing, and updating requirements; data and data element definition, message structure and flow, operational sequence of events; and error detection and recovery procedures. The ICD does not explicitly prescribe the underlying logic or behavior that actually generates the variables that should go back onto the interface.

An assumption underlying the use of the ICD is that data and control coupling between components are minimized and that if a change to one function does not require a change to the ICD, then the functions using those parameters cannot be impacted by the change. This assumption is false, as shown in an example involving flaps control provided by Bartley and Lingberg.

Figure E.1 shows the interface between the flaps system controller in which a discrete variable of "flaps extended" is passed between supposedly independent avionics functions. Without further definition of the exact technical meaning of flaps extended, a true state of the flaps-extended discrete variable could possibly mean any of the following:

- Both left and right trailing-edge (TE) flap surfaces are detected in the 1 or greater flap detent.[38]
- Both left and right trailing-edge flap surfaces are not detected in the "up" flap detent.
- The flap lever handle is detected in the 1 or greater flap handle detent.
- The flap lever handle is not detected in the up-flap handle detent.

All four of these possibilities have differing characteristics from each other, and different receiving functions might use the flaps-extended discrete in a different way. For example, once the trailing-edge flaps begin to move, the logic that determines "flaps not in the up position" will be satisfied almost immediately. In contrast, the flaps may take 5 to 10 seconds to fully reach the "flaps detected in the '1' detent" position. Furthermore, if the flap surfaces will not respond to a valid command due to a hydraulic system failure, the flap lever position will no longer reflect the true position of the flap surfaces once the lever is moved out of the up detent. Obviously, how a signal is computed needs to be

[36] Even for these systems, results from combining component analyses can be incorrect when subtle indirect interactions among components distort the combined component interactions.

[37] Gregg Bartley and Barbara Lingberg, Certification Concerns of Integrated Modular Avionics (IMA) Systems, Federal Aviation Administration, Washington D.C.

[38] A detent is a device used to mechanically resist or arrest the rotation of a wheel, axle, or spindle.

understood by the designers of any function that uses that particular signal. Differences such as those described may be of critical importance to using the system.

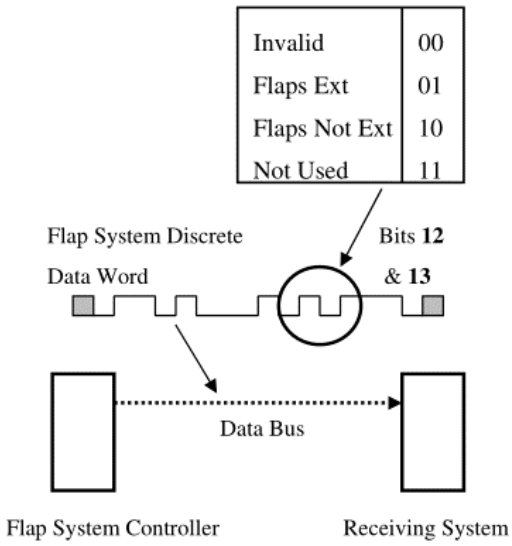


Figure E.1: Flaps-extended (Ext) discrete data transmission [3].

If only the information in Figure E.1 is communicated via the data bus, then a change in the logic of the flap system controller may negatively impact the behavior of the function using the flaps-extended variable. The receiving system(s) developers will thus not be privy to the change, and the system-level impact will not be understood. For example, a problem discovered later in the development cycle, after the design of the individual applications have been ”finalized,” may lead to a change in the logic of the flap system controller.

One potential method of addressing this problem with the flap alert logic is to compute the flaps-extended variable using the flap lever position instead of the actual flap surface position. The physical meaning of the flaps-extended variable has changed.

According to Bartley and Lingberg:

> The point illustrated is that it cannot not [sic] be left to the designers and system experts of the function being updated (in this example, the flap system) to determine the effect of that change on downstream users. The experts that are required to assess the impact of the change are the designers and analysts of the using system, not the source system. A Change Impact Analysis that crosses functional boundaries is clearly required for this example, even though the ICD for the source function did not change. Additionally, this example illustrates why robust partitioning does not totally insulate functions residing in one partition from changes to a function in a different partition.

Traditional hazard analysis techniques such as FMEA and fault tree analysis focus on individual component failure or faulted modes. This focus on component reliability assumes that robust partitioning and interface control are valid and that components do not interact either directly or indirectly. However, as the example illustrates, these assumptions usually do not hold for complex systems.

The problem can be theoretically explained and solved using STPA.

Appendix F: Basic Engineering and System Engineering Concepts for Non-Engineers

In reviews of an early draft of this handbook, we discovered that we had assumed some engineering background that not all readers had: Users of STPA are not necessarily trained in engineering. This appendix provides an introduction to basic engineering concepts that may be unfamiliar to some users of this handbook.

I have attempted to write the sections in this appendix to be independent so that readers can pick and choose sections that interest them. No assumption is made about any particular educational background in covering these topics, which include:

- What is a "system"?
- Basic systems engineering concepts and terminology
- The concept of "control" in engineering
- Systems theory vs. complexity theory

What is a "System"?

The most basic concept underlying everything else in this handbook is that of a *system*.

> *System*: A set of things (referred to as system components) that act together as a whole to achieve some common goal, objective, or end.

Some definitions of a system leave out the "goal" or "objective" and state essentially that a system is a connected set of things or parts forming a unified whole. This definition does not capture the way the term "system" is usually used. A shoe, a star, and a locomotive are a set of things or potential parts of a system but most people would not normally consider this to be a system. It could be considered as a system if a purpose could be conceived for considering these individual things together; the purpose is basic to the concept.

Other definitions state that the components of the system must be interdependent or connected or interacting, but none of these conditions are really necessary to have a system and constrain the definition in a way that excludes things that usually are considered to be systems. The system components may be either directly or indirectly connected to each other, with the latter including connections involving the system purpose only and various types of non-linear interdependencies.

The consequence of this definition is that a goal or objective for the system is fundamental. But there may be different objectives for those defining and viewing systems. An airport is used throughout this section to illustrate various points. To a traveler, the purpose of an airport may be to provide air transportation to other locales. To local or state government, an airport may be a means to increase government revenue and economic activity in the area of the airport. To the airlines, the purpose of an airport may be to take on and discharge passengers and cargo. To the businesses at the airport, the purpose is to attract customers to whom they can sell products and services. Therefore, when talking about a system, it is always necessary to specify the purpose of the system that is being considered.

A system is an abstraction, that is, a model conceived by the viewer.

The observer may see a different system purpose than the designer or focus on different relevant properties. Specifications, which include the purpose of the system, are critical in system engineering. They ensure consistency of mental models among those designing, using, or viewing a system, and they enhance communication. Notice that different components of the airport may be included in a

particular “airport” system, such as passenger check-in counters, ramps to the planes, and taxiways in the airline view of an airport versus shops and customers from the commercial view. Note again that these are models or abstractions laid upon the actual physical world by human minds. The components that are considered in any “airport system” or subsystem and the role they play in the system as a whole may be different for each concept (model) of an airport system or of airport subsystems. More on this topic can be found in the next section on the importance of specifications in systems engineering. The basic idea here is that the purpose or goals of the system being considered must be specified and agreed upon by those modeling and analyzing a particular system and that these aspects of systems are abstractions or models imposed by the viewer on the real world objects.

For engineered (man-made) systems, the purpose is usually specified before creating the system although observers of the system may interpret the purpose differently than originally intended by the designers. For natural systems, behavior may be interpreted as purposeful by the observers of the system.

There are some basic assumptions underlying the concept of a system:

- The system goals can be defined
- Systems are atomistic, that is, they can be separated into components with interactive behavior or relationships between the components whether the interactions are direct or indirect.

Systems themselves may be part of a larger system or be divisible into subsystems (viewed as components of the overall system). Note that the definition here is what is called “recursive” in mathematics and computer science. That is, the definition is made in terms of itself. A system is usually part of a larger system and it can be divisible into subsystems. Figure F.1 shows a typical abstraction hierarchy for a system labeled A (for example the system “airport”) and for three subsystems, which in turn can be conceived as systems themselves.

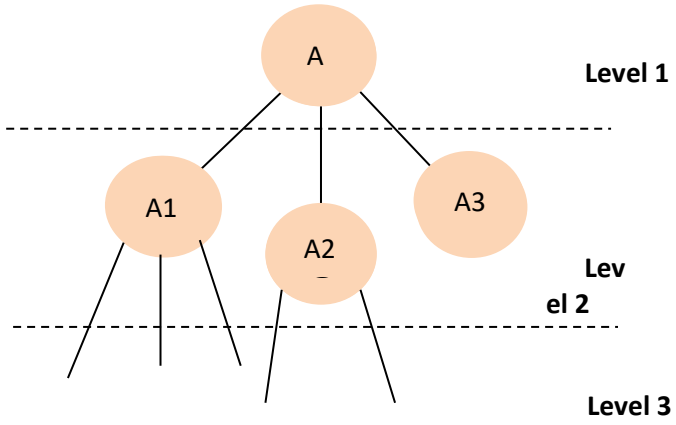


Figure F.1: The abstraction System A made be viewed as composed of three subsystems. Each subsystem is itself a system.

Figure F.1 is not a connection diagram. Rather it is a hierarchical abstraction that shows the different views of a system at different levels of abstraction or modeling. Figure F.2 shows an abstraction where A itself is conceived as part of a larger system AB.

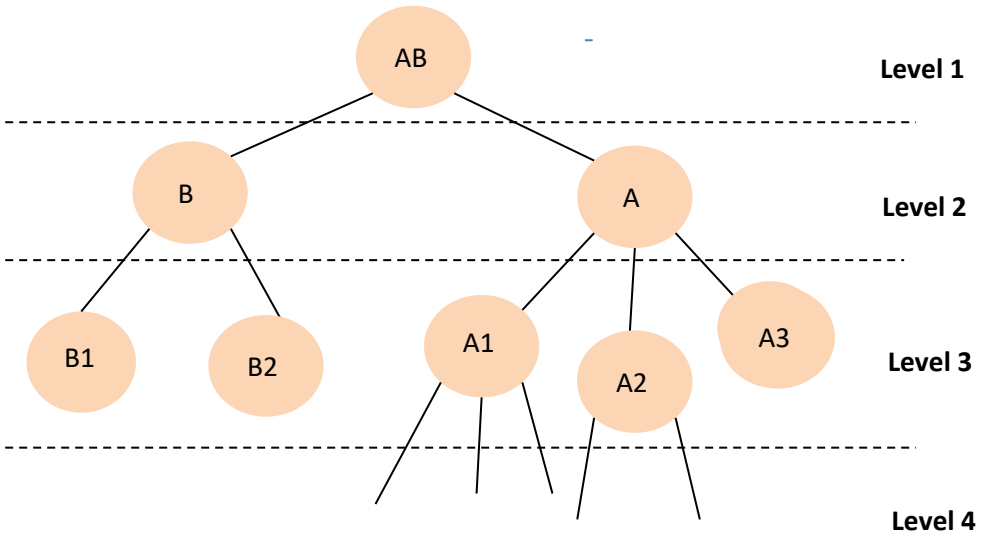
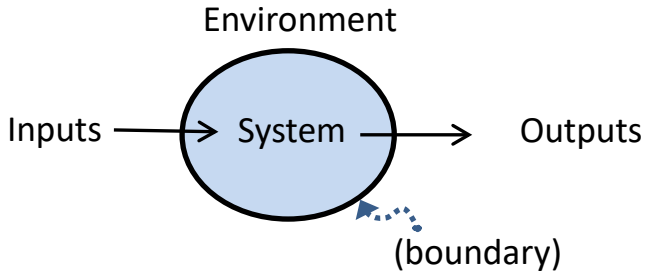


Figure F.2: System A can be viewed as a component (subsystem) of a larger system AB

The recursive nature of the definition of a system is important because many people have suggested that "systems of systems" must be treated differently than systems. In fact, the same general system engineering and analysis methods and techniques are applicable to all systems. A "system of systems" is just a "system" using the definition of a system as defined in system theory and as the concept is used in engineering.

Systems have *states*. A state is a set of relevant properties describing the system at any time. Some properties of the state of an airport viewed as an air transportation system may be the number of passengers at a particular gate, where the aircraft are located and what they are doing (loading passengers, taxiing, taking off, or landing). The components of the state that are relevant depend on how the boundaries are drawn between the system and its environment.

The *environment* is usually defined as the set of components (and their properties) that are not part of the system but whose behavior can affect the system state. Therefore the system has a state at a particular time and the environment has a state. The concept of an environment implies that there is a boundary between the system and its environment. Again, this concept is an abstraction created by the viewer of the system and need not be a physical boundary. What is part of the system or part of the environment will depend on the particular system and its use at the time.



System *inputs* and *outputs* cross the system boundary. This model is usually called an *open* system. There is also a concept of a *closed system* (which has no inputs or outputs), but this concept does not have much relevance for the engineered systems with which we are most concerned in system safety.

What is Systems Engineering?

System engineering is the attempt to put structure into the design and construction of a system in order to improve the results of the engineering effort. For relatively simple systems, system engineering may not be necessary. Systems engineering came into being after World War II when we started building significantly more complex systems, especially missile and other defense systems, and found that an informal design and construction process often resulted in systems that did not satisfy the goals envisioned for the system and the constraints on its potential behavior, i.e., constraints how it achieved the goals. Lack of a structured process also led to projects being late and over budget.

System engineering is a large subject, but only two concepts are necessary to understand this handbook. The first is the concept of the system engineering phases and process. The most basic model of these phases is shown in Figure F.3.

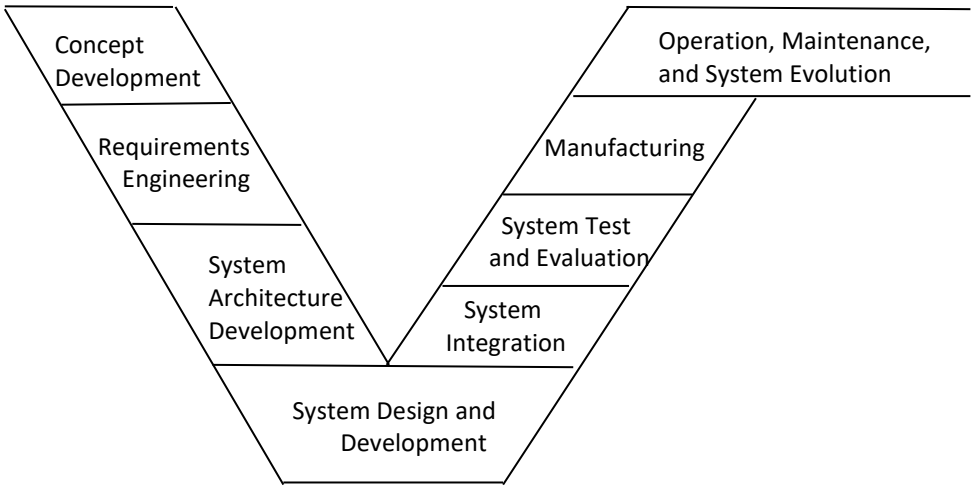


Figure F.3: The Basic System Engineering "V-model"

This model is called the V-model, although different labels may be placed on the various components. Sometimes the phases are drawn in a straight line (if the line tilts downward it is called the "waterfall" model), but there is no difference except in the way it is drawn. The basic idea is that the best results ensue if a structured process is used, where the system concepts are first developed and the basic goals and constraints on the system are defined. There may be various feasibility and other analyses done at this early time in development to stop the process from going down paths that require retracing later. Requirements engineering is usually the next step in the process where detailed requirements are developed for the system given the basic goals and constraints earlier agreed upon. System architecture development means that a basic architecture or high-level design of the system is created before detailed design and development. The left side of the "V" represents the basic design process.

The right side of the V shows the later stages in development when assurance and manufacturing take place. Sometimes operation of the system is included (which is then called a life-cycle model)—as

was done in Figure F.3 and Chapter 3 of this handbook because safety analysis needs to continue through the entire life of the system. In right side of the V-model process, the various components that have been designed and constructed are integrated and go through a testing and evaluation phase. They are then manufactured and used.

The figure above is purposely simplified to reduce clutter and only reflects the overall process components. There is always lots of bouncing around the phases and not one straight line process. Requirements are never totally specified at the beginning, for example, as the design process itself will generate more understanding of the problem being solved and therefore result in adding to or changing the original requirements and constraints. Serious problems during development may occasionally require the project to revert totally to an earlier phase. But this step-wise process is generally followed.

In addition, this model should not be taken too literally. The labeling of phases does not mean that some parallelism is not possible and, in fact, common. Waiting for all testing to take place, for example, until the end creates grave risks to a project. Testing of various kinds may start in the earliest concept phase (perhaps in the form of simulations or prototyping) and usually continues through every phase. Also, various labels may be applied to the phases, depending on the type of system being designed. The goal is simply to define a structured process that will be tailored for the various needs of specific projects. Chapter 3 shows how STPA can generally be used in the various phases and activities involved in engineering a complex system.

Another aspect of the V or waterfall model is that it is documentation driven. In my experience, the key to successfully designing a complex system lies in specification. There are usually large numbers of people involved in such systems (sometimes in the thousands) and development and operation occurs over what can be a very long time frame. Specification is necessary to ensure that the result satisfies the needs of the users and that decision making does not descend into chaos because of communication problems.

Specification starts in the earliest stages of development. The previous section of this appendix defined a system as an abstraction whose purpose is imposed by the viewer of the system. In order to ensure that everyone agrees on the basic objectives of the system being created, specifications must at the very least specify the system purpose (goals), the relevant system properties of interest or desired, and any constraints on how the goals can be achieved. In addition, the system boundary must be identified (i.e., what is within the boundaries and under the control of the designers and what is part of the environment that is assumed by the designers and not under their control). Other aspects of the initial specification include the inputs and outputs, the physical or logical components, the structure, the relevant interactions among components and how their behavior affects the overall system state. As development proceeds, the initial specification will be refined and details developed.

Traceability is another important concept in system engineering. Very large and complex systems may have thousands of pages of detailed specifications by the time they are completed. Showing the relationship between parts of the system and what assumptions or previous results of the development process particular decisions were based on is a requirement for creating and operating the system. For example, in STPA, unsafe control actions are traced to system-level hazards and scenarios traced to the unsafe control actions they cause. When changes need to be made, it is possible to determine what needs to be changed without starting over from scratch. Traceability is also a way to record some of the types of rationale that underlie decision making.

The Concept of "Control" in Engineering

The concept of control is important in the conception, design, and operation of a system. A system can conceivably operate without any type of control over its behavior, but it would be difficult to ensure

that the system satisfied its goals while at the same time constraining the way it achieves those goals so that adverse, unwanted consequences do not also result.

Control is basic to engineering, especially complex systems. Engineers are dealing not with living things, but with man-made things —automobiles, dams, aircraft, chemical plants, spacecraft, washing machines, robots—that are built to accomplish some goal. If they are not inert (like a bridge), there is a need to operate or control their operation, i.e., the states that they get into.

Non-engineers sometimes interpret the word "control" differently than the way it is used in engineering. Control is not about forcing individual compliance to specified roles and procedures through some type of militaristic-style authority. Rather, control is being used here in a very broad sense and may be implemented through design controls like interlocks and fail-safe design, through process controls such as maintenance and manufacturing procedures, or even social constructs such as culture, incentive structures, and individual self-interest. Without the imposition of some form of control, there would be no way to ensure that the system goals and constraints are achieved.

Simple design features acting as controls often can improve safety without also increasing complexity and cost, but this usually requires considering safety early in the design process. The most effective way to increase safety through design is to use a design that eliminates hazardous states. Next best is to design such that hazards are rare and easy to handle if they do occur. A design that reacts to the occurrence of hazards by minimizing the resulting damage is clearly the least desirable form of hazard control but often such controls must be included.

A control loop is thus only one form of control. It is emphasized in *Engineering a Safer World* because the other aspects of basic design for safety were covered so extensively in my previous book titled *Safeware*. In addition, system theory emphasizes the concept of the control loop or active control. It is not possible to teach the techniques used to design for safety in this appendix, but some basic concepts such as the distinction between p*assive* and *active* control may be helpful in understanding general safety design techniques.

There are three types of design techniques to "control" safety. The first is to use an intrinsically or inherently safe design. An intrinsically safe design is one that is not capable of producing the hazard of concern. For example, the system design may not have sufficient energy to cause an explosion. If the concern is with the use of potential toxic substances, there may be non-toxic substitutes. If the potential exists for lost information over a communication network, then more direct communication may be used if it is feasible.

If an intrinsically safe design is not possible, the use of passive control mechanisms may be possible. Passive controls do not require a positive action in order to be effective, but instead rely on basic physical principles, such as gravity. A physical interlock that prevents the system from entering a hazardous state without any overt control action is a passive control mechanism. Examples include:

- A pressure sensitive mat or light curtain that shuts off power to a robot or dangerous machinery if someone comes near.
- A physical device that ensures the correct sequencing of valve turn-off and turn-on or ensures that two valves cannot both be on or off at the same time.
- A freeze plug in a car's engine cooling system where expansion will force the plug out rather than crack the cylinder if water in the block freezes.
- A fusible plug in a boiler that becomes exposed if there is excessive heat and the water drops below a predetermined level. In that event, the plug melts and the opening permits the steam to escape, which reduces the pressure in the boiler and prevents an explosion.

- A speed governor that does not allow speeds over a specific limit or relief valves that maintain pressure below dangerous levels.

Passive controls are often designed so that the system essentially fails into a safe state. In fact, the basic approach to aircraft safety is to build the system to be fail safe. Some examples of fail-safe design are:

- Old railway semaphores that used weights to ensure that if the cable (controlling the semaphore) broke, the arm would automatically drop into the STOP position.
- Air brakes that are held in an off position by air pressure. If the brake line breaks, air pressure is lost and the brakes are automatically applied.
- A deadman switch
- A bathyscope where ballast is held in place by magnets. If electrical power is lost, the ballast is released and the bathyscope ascends to the surface.
- Designs where the effects of a failure (such as fan blades being ejected when an aircraft engine fails) are contained so that adjacent components are not affected.

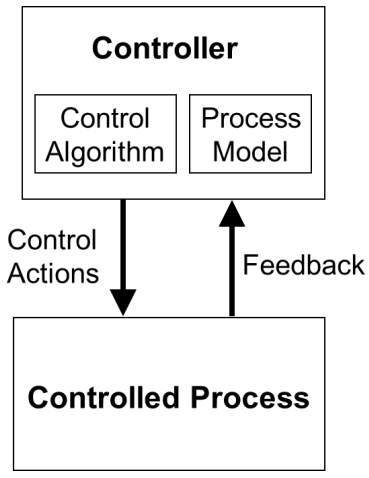
An *active control*, in contrast, requires a hazardous condition to be detected and corrected. Active controls today often involve the use of computers and the basic control loops described and used throughout this handbook. Notice that there is more that can go wrong with active controls, e.g., the failure or hazardous state may not be detected or it may be detected but not corrected or it may not be corrected in time to prevent a loss. Passive controls are more reliable as they usually depend on physical principles and simpler designs while active control depend on less reliable detection and recovery mechanisms and usually much more complex designs. Passive controls, however, tend to be more restrictive in terms of design freedom and are not always feasible to implement in complex systems.

Active controls usually involve a control loop.

What is a Control Loop?

A control system commands, directs, or regulates the behavior of other devices or processes using control loops. Such control systems may range from a single home heating controller using a thermostat to control a boiler to large industrial control systems used for controlling complex processes and machinery.

A very basic control loop is shown below (and elsewhere in this handbook):



In order to control a process, the controller must have a *goal* or goals, which can include maintaining constraints on the behavior of the controlled process. In addition, the controller must have some way to

affect the behavior of the controlled process, i.e., the state of the controlled process or system. These are labeled control actions in the figure. In engineering, control actions are implemented by *actuators*. In order to know what control actions are necessary, the controller must be able to determine the current state of the system. In engineering terminology, information about the state of the system is provided by *sensors* and is called *feedback*. Finally, the controller must contain some model of the state of the controlled process as explained elsewhere in the handbook.

As an example, a simple thermostat may have a goal of maintaining a temperature within a specific range. Feedback provides information to the controller about the current state of the controlled process, in this case the temperature of the air in the room. The feedback is used to update the controller's model of the process. Other information may also be part of the process model, such as environmental information (outside air temperature) or basic information about natural temperature fluctuations. The controller uses the process model to decide what control actions to provide, e.g., apply hot air or cool air, in order to change the state of the room temperature (controlled process) to stay within the required range.

There are two general operational modes for a control loop: feedback control and feedforward control. *Feedback control* is what was described in the paragraph above. When driving, a driver may read the speedometer (feedback) and decide to brake or step on the accelerator to keep the automobile's speed at a desired level.

An example of *feedforward control* occurs when the driver approaches a hill. The driver could wait until the car starts to slow down and then step on the accelerator, but more likely an experienced driver will anticipate that acceleration will be required to maintain a constant speed when going up the hill and the driver will increase acceleration before the car actually slows down. Thus, in feedforward control, the controller uses a model of the current state of the process (in this case the car speed) and the future (operating on an incline) and then provides a control action without specific feedback to identify the need.

Often feedback and feedforward control are used in tandem. In our example, the driver might anticipate the need to step on the accelerator before feedback is received using feedforward control, but feedback can be used to adjust the feedforward control actions so that a constant speed is maintained on the incline if the feedforward calculations are not perfect.

In the chapter of this handbook on leading indicators, *shaping actions* are described as attempts to maintain a safe state by designing controls to maintain assumptions, prevent hazards (unsafe states in the controlled process), and control migration of the process to states of higher risk. Shaping actions provide a type of feedforward control. The type of control used may be passive or active. *Hedging or contingency actions* provide a type of feedback control as they involve monitoring the system for signs of increasing risk and responding accordingly. Hedging actions that include monitoring the effectiveness of the shaping actions provide a combination of feedforward and feedback control.

The engineering concept of control (which is based on system theory) can also be applied to social and organizational systems and has been used extensively in management theory and social sciences.

Systems Theory vs. Complexity Theory

STAMP is based on systems theory, not complexity theory. Contributions to systems theory have been made by many people, but Ludwig von Bertalanffy, an Austrian biologist, is credited as one of the founders of what he called general systems theory. Norbert Weiner explored the implications for mathematics and engineering. Weiner called his theory *cybernetics* but that term has faded over time and Bertalanffy's terminology is generally used today.

Systems theory, as you have seen in this handbook and perhaps elsewhere, is a set of principles that can be used to understand the behavior of complex systems, whether they be natural or man-made systems. *Systems thinking* is the term often used to describe what people are doing when they apply systems theory principles. These principles are briefly described in Chapter 1 of this handbook.

Complexity theory grew out of systems theory and other concepts in the 1960s and is usually associated with the Santa Fe Institute and researchers working there. Some commonalities exist between Systems Theory and Complexity Theory in that both include terms like emergence and focus on complex system behavior as a whole rather than on reduction or decomposition into components. The basic components of system theory are emergence, hierarchy, communication and control and these also are included in complexity theory. Both systems theory and complexity theory also include concepts of feedback and feedforward control, adaptability, nonlinear interactions, and constraints. Both reject reductionism or decomposition as a principle for understanding system behavior.

There are, however, significant differences. Complexity theory was created to describe natural systems where seemingly independent agents spontaneously order and reorder themselves into a coherent system using laws of nature that we do not yet fully understand. Systems theory, in contrast, is more appropriate for man-made and designed systems where the system is purposely created by humans using some engineering or design process and where the basic design is known and changes are controlled.

Another major difference is that systems theory considers all systems as displaying emergent behavior while complexity theory divides systems into four types: simple, complicated, complex, and chaotic, each with different degrees or types of emergence.

As such, systems theory appears to be most appropriate for engineered or designed systems while complexity theory is most appropriate for natural systems where the design is unknown, such as the weather, and for sociological systems, such as communities, that are not designed and where there is a lack of order and it is very hard or impossible to predict the emergent behavior. Complexity theory deals with behavior that cannot be designed but instead must be experienced and studied as it arises.

Systems theory, at least for me, is easier to understand and use. Complexity theory frameworks can be difficult and confusing to apply and understand and may be overkill for the engineering goals related to improving safety and other emergent system properties. Thus, systems theory was selected as the basis for STAMP.

Appendix G: A Control Model to Assist In Causal Scenario Generation

The goal of STPA is to identify the causal scenarios that can lead to a hazardous state. A model was provided (see Figure G.1 below) in *Engineering a Safer World* that showed some of the general reasons why hazardous states might be generated. The goal of providing this model was to assist in generating scenarios. One problem, however, is that the model is very abstract and leaves out details that can be useful in scenario generation. The model also combines automated and human controllers in one general box labelled "Controller," another abstraction that leaves out important differences. Later, in Chapter 9 of the book, another model was created to describe design techniques. This model, too, is overly general. Since the book was published, other more detailed models for human controllers have also been generated. In this appendix, we describe a model that is more complete and hopefully more useful to those identifying causal scenarios.

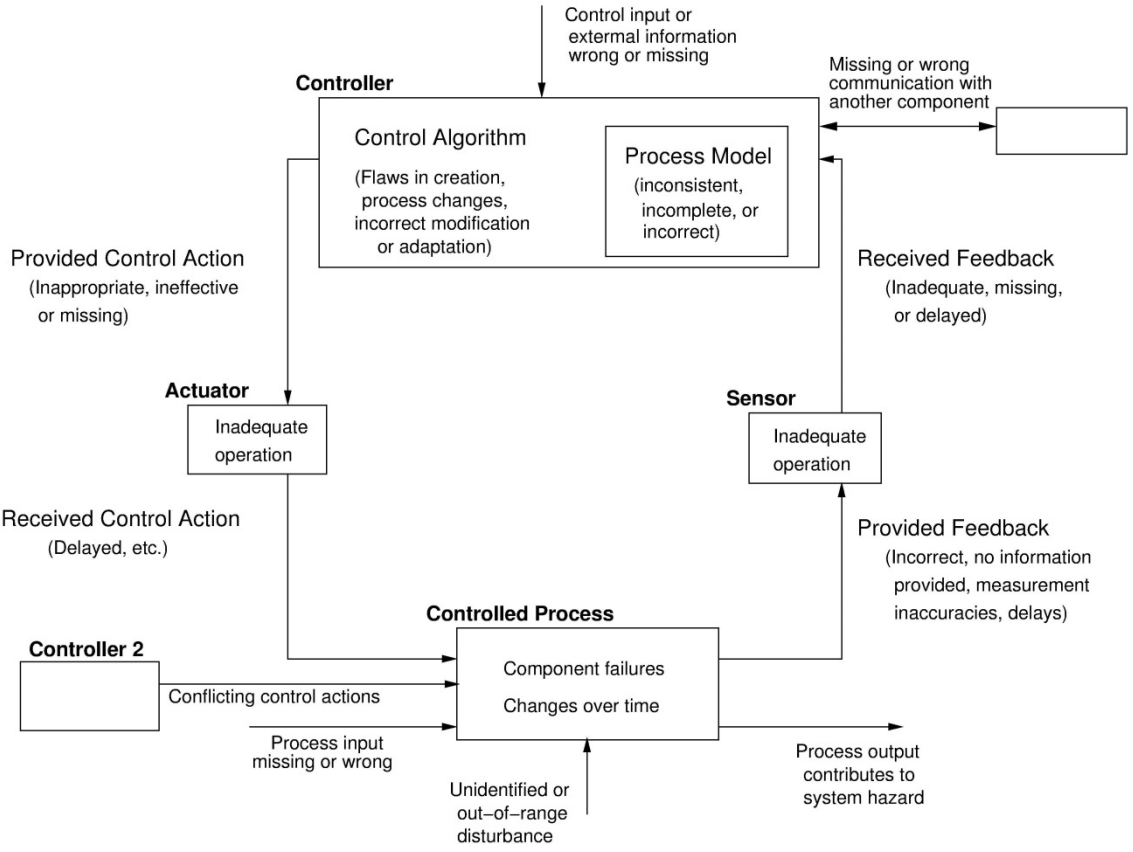


Figure G.1: Previous General Model to Assist in Generating Causal Scenarios

An Abstract Model for Causal Scenario Generation

The new model is shown below in Figure G.2. The parts of the model are then described along with how they may be used to create scenarios. There are too many potential causal factors to include them all on the general model diagram itself so they are described separately. When generating causal scenarios using this model as a guide, you will start from a specific control model for your application (system). The generic components below will correspond to the parts of your model. The generic guidance can be used to assist you in generating the specific causal scenarios for your system.

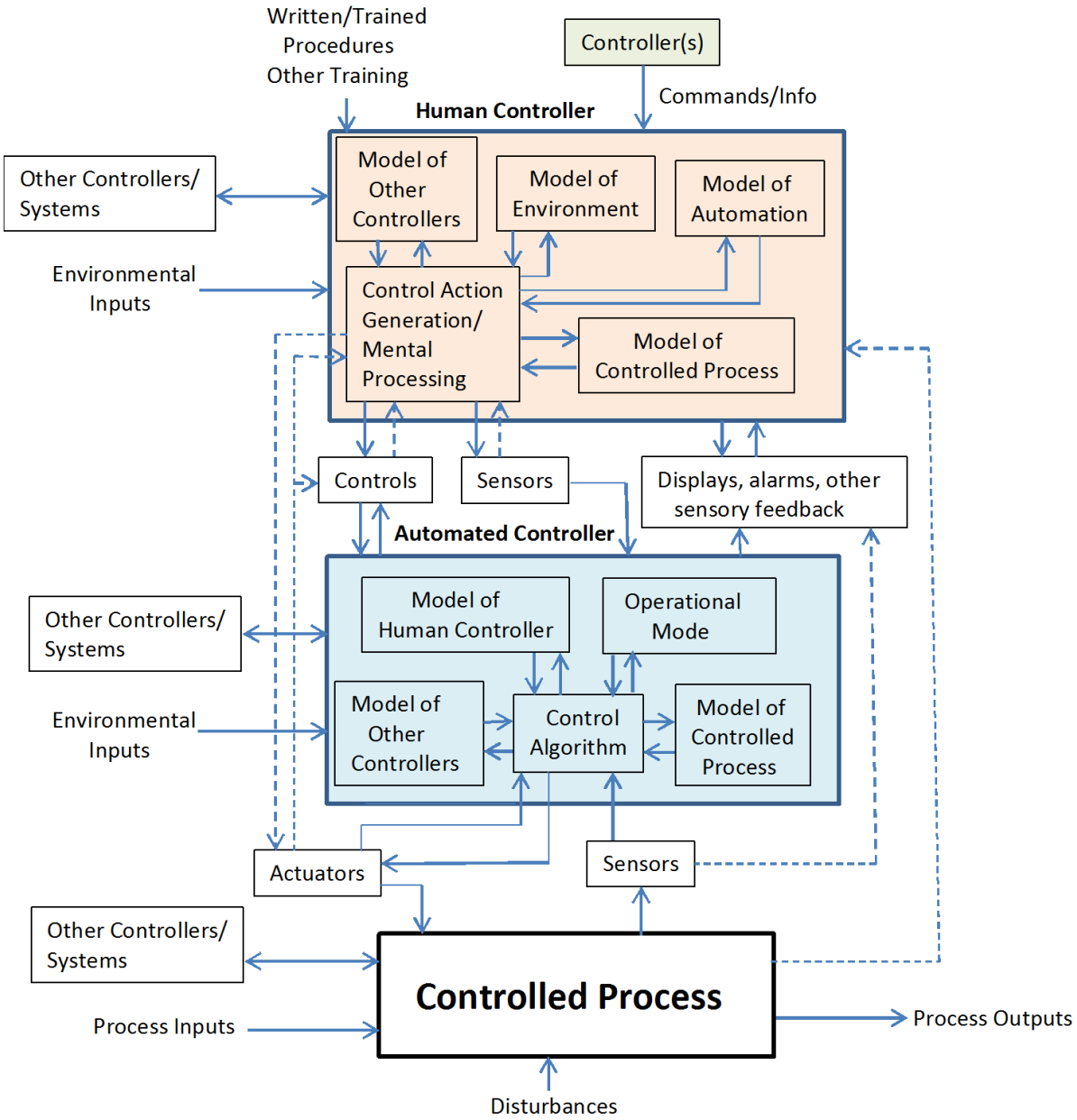


Figure G-2. New General Causal Control Model for STPA Causal Analysis Scenario Generation

Controlled Process

A hazard is defined in terms of the state of the controlled process at the bottom of the figure, e.g., the attitude of the aircraft, the speed of the automobile, the position of the robot. States are made up of components or variables. As the goal of STPA is to identify how the controlled process gets into a hazardous state, then we need to look at the ways the controlled process state can change state. Anything that can change that state is potentially part of a causal scenario for a hazard.

Failures or Degradation over Time of the Controlled Process Components

Failures of the controlled process hardware can cause it to get into a hazardous state. Changes over time of that hardware, such as corrosion, might make it operate differently than designed. Controls such as hardware interlocks may also fail or degrade in such a way that a hazardous state is created. Remember, the goal is not to do a FMEA, that is, to look at all possible failures and their results. The goal is to start from the hazard and determine what conditions in the controlled process could contribute to that hazardous state. As an example, let's assume that the hazard of concern to the causal analysis is the loss of pitch control of an aircraft. Failure of the aircraft pitch control devices, such as the slats, flaps, ailerons, and elevator, could lead to the hazard. If the hazard is a lack of effective braking (deceleration), that may be due to insufficient hydraulic pressure (pump failure, hydraulic leak, etc.).

External Disturbances

Disturbances to the process can lead to a hazardous state, such as lightning affecting an aircraft's electrical system. Or a bird is sucked into an engine and affects the power (thrust) of the aircraft. The national airspace may be disrupted by weather that increases the hazardous state for individual aircraft currently in that airspace. The environment may interfere with the proper execution of the controlled process, such as inadequate braking due to a wet runway (e.g., the wheels hydroplane).

Direct Inputs to the Controlled Process

Direct inputs to the process may affect its state. For an aircraft, loading and unloading passengers or cargo change the weight, which can affect the controllability of the aircraft. For the national airspace, inputs are aircraft that enter the airspace and outputs are those that leave it. If the components of the controlled process hazardous state are affected by any inputs (or outputs), then those process inputs and outputs must be considered in causal scenario generation.

Controllers of the Controlled Process

Clearly, the controllers of a hazardous process can potentially contribute to a hazardous state. The model shows the primary controllers in the blue and tan boxes. In an aircraft, these may be the human pilots and automated flight control system(s). Note that the model does not imply that the pilot(s) are physically onboard the aircraft. The model in Figure G.2 shows another box (to the left of the controlled process) representing other controllers or systems that can impact the state of the controlled process or its components. For example, maintenance activities or the lack of them can contribute to hazards in physical systems. The impact of other automated and human controllers is described separately below.

Automated Controllers

Only one automated controller is shown in the diagram (the blue box) although there may be several of these and they may be designed in a hierarchical structure. Because the model for automated and human controllers always have the same components, only one is shown here.

Except in very simple systems, automated controllers provide control actions to actuators, which change the controlled process state. Sensors provide feedback about the current process state.

Control Path from the Automated Controller to the Controlled Process through the Actuator

As stated in Chapter 2 of this handbook, this control path transfers control actions from the automated controller to the controlled process. The control path may consist of a simple actuator, may involve a series of actuators, or it may transfer control actions through a complex network with switches, routers, satellites, or other equipment. No matter how it is implemented, problems along this control path can lead to hazardous controlled system states when control actions needed to prevent a hazard are not executed, are improperly executed, or are delayed to the point where they become hazardous under some conditions. Examples include actuator failure, transmission problems along the control path, or delays in sending or executing the control action. For hazards related to security flaws, an adversary may impede a control action required to prevent a hazard or inject a control action that leads to a hazard in the controlled process.

Feedback Path from the Controlled Process to the Automated Controller through Sensors

As with the actuator control path, flaws in the feedback path may stem from transmission problems including delays, measurement deficiencies and inaccuracies, sensor failure, or other flaws related to sensor design or operation. Again, an adversary might negatively impact the feedback path. If a flawed process model of the automated controller can contribute to an unsafe control action, then the feedback path should be considered in the causal scenario generation. For example, data may be delayed to the point where it no longer reflects the current state and the controller acts on incorrect information. Do not stop with "flawed process model" in your causal analysis generation. It does not provide you with enough information to provide safeguards against the unsafe control action. Instead, you need to determine why the process model could be flawed.

Automated Control Algorithm

The automated control algorithm has two primary functions: (1) generate control actions and (2) maintain accurate information (models) about the state of the controlled process and external system components and environment that can impact the generation of control actions. Flaws in either of these functions can lead to UCAs.

1. Generating Control Actions

The automated control algorithm may generate unsafe control actions due to flaws in the control algorithm itself, unsafe commands provided to the automated controller from an external source, or flaws in the models that the automated controller has of the controlled process and its environment. The causes of the latter are described in the next section.

STPA generates the safety requirements and constraints for the automated control algorithm. These must, of course, be verified to be correct. A causal factor here, then, is inadequate communication and verification of the operational safety requirements and constraints for the control algorithm.

2. Maintaining Internal Models of the Contextual Factors Associated with UCAs

The safety of decision making about what control actions to perform can also be impacted by the information that the automated controller uses to make those decisions, even if the actual automated algorithms satisfy the safety requirements and constraints. The automated controller uses four types of information in that decision making: (1) the state of the controlled process, (2) the state of other controllers of the controlled process, (3) the operational mode of the automation, and (4) the state of the human controller (in some sophisticated new systems). Various types of flaws in these models can lead to unsafe control actions by the automated controller. The context for the UCAs will contain the information about what flaws in the model can lead to the UCA. For example, the BSCU Autobrake provides a brake control action during a normal takeoff. This UCA then can occur when the BSCU is confused about the state of the controlled process, that is, it does not know that the aircraft is in a

normal takeoff state when a brake action command is given. The causal scenarios will include the reasons for the BSCU Autobrake being confused about the aircraft state and the reasons that a brake command might be given under these conditions.

Model of the Controlled Process

The model of the controlled process is the state that the automated controlled thinks that the controlled process is in. This model may include information about the environment in which the controlled process is operating. The model is updated by the controller, which we will define as part of the control algorithm (if it is separate, that won't matter[39]). The automated control algorithm gets direct information about that state from sensors that are designed to measure controlled process variables. There may also be designs where the process model is automatically updated without going through the control algorithm. An example is the loading of specific flight data right before launch of the Space Shuttle. Missile systems also do this. Appendix B of *Engineering a Safer World* details a very costly accident arising from errors (typos) in a software flight data load.

If the control algorithm is separated into several pieces (such as handling of regular braking and automated braking), there are various ways that updating of the controlled process model may be flawed through unknown or unconsidered interactions among them. These cases should be considered when identifying scenarios that lead to UCAs.

In updating the controlled process model, the control algorithm may receive indirect information about the state of a controlled process or make assumptions about it. One common assumption is that when the control algorithm issues a control action, it may assume the action is executed and update its process model. For example, the Electronic Flight Control System on an aircraft issues a control command for an actuator to move the elevator a certain number of degrees. It could then assume both that the elevator has moved and that it has moved by the specified amount. Reasons for the commanded movement not occurring could involve flaws in the communication links to the actuators, inadequate actuator operation, and failures or faults in the controlled process (the elevator itself). Feedback about the non-execution of the command may not be included in the system design or may be included in the design but not received or not processed by the automated controller. Various types of technical design issues may also be involved such as latency (for example, the time between when an input is received and when it is processed, which will be impacted by software architecture design decisions such as the use of interrupts or polling). Data age problems have also led to hazards, that is, data is used that is no longer valid or output commands are delayed in their execution to the point where they are hazardous. The latter can occur because of delays in the actuation path or because of conditions in the controlled process. For example, on one military aircraft, a maintenance worker was working on the bomb bay door and activated a mechanical interlock to prevent the door from closing while he was in its path. At the same time, other people working in the cockpit issued a command to close the bomb bay door. The command did not execute (because of the mechanical interlock) until several hours later when the maintenance worker released the mechanical interlock. The worker was killed.

There are many other reasons for inconsistencies between the process model and the actual process state (and between models in various controllers). One possible reason is delays in updating process models, which can lead to inconsistencies and unsafe control actions. In initial startup, designers may have assumed default values that are not correct in some cases. For example, some aircraft automation

[39] It may be useful to separate the various functions of the automated controller software, but they are combined here. One reason to separate them is that the design of the software will be easier to implement or change if it is separated logically. It might also be easier to verify that the software is correct.

is supposed to be powered up and initialized before takeoff. Default values are often provided for that case. However, if the device is started up later than takeoff, the default values may not hold. TCAS, for example, assumes that it has been started while the aircraft is on the ground and initializes the system with values corresponding to being on the ground. The same is true of process models for devices that are shut down for maintenance and restarted with perhaps an assumption that the controlled process state is the same as when the system was shut down for the maintenance procedures. Another example is that human controllers may decide to restart a device when it does not appear to be working correctly. TCAS, for example, allows pilots to reboot it in the air. The device, however, restarts with initialization values that may not be correct at that point in the flight. Input to update the process model may also arrive before the device is powered up, after shut down, or while the device is disconnected (off-line) and therefore may be ignored.

Model of the Operational Modes

The model of the operational modes may also be related to the context defined in the UCAs. There are four types of modes that must be considered: (1) the controlled process mode, (2) the automation mode, (3) the supervisory mode, and the display mode.

1. Controlled Process Mode

The controlled process mode identifies the operating mode of the controlled process. For example, an aircraft (controlled process) may be in takeoff mode, landing mode, or cruise mode. The requisite modes to model will depend on the contexts identified for the UCAs. For example, if the aircraft is in cruise mode (rather than in landing mode), operation of the reverse thrusters can be unsafe. That context should be identified in the UCA context table (i.e., operation of the reverse thrusters when the aircraft is not in landing mode). The reasons (causal scenarios) for the UCA will be at least partially related to the automation being confused about the current mode of the controlled process.

2. Basic Automation Operating Mode

In addition to the controlled process, the automation itself (automated controlled in Figure G.2) has modes of operation. Those modes will often be used in the logic of its control algorithm and affect its outputs. Examples include nominal (normal) behavior mode, shutdown mode, and fault-handling mode. As an example, the automation may be in partial shutdown mode after detecting a fault in itself or in the controlled process. In the Air France 447 loss, the autopilot shut itself off (went into shutdown mode) when a pressure probe (pitot tube) on the outside of the aircraft iced over and the autopilot could no longer tell how fast the aircraft was moving. The fly-by-wire system, at the same time, continued operating but switched into a mode where it no longer provided protection against aerodynamic stall. On modern Airbus aircraft, operational modes of the electronic flight control system include *normal*, *alternate*, *direct*, and *mechanical laws*, each of which changes the behavior of the automated flight control system.

Partial shutdown modes in automated controllers can also be very confusing to human controllers, which will be covered later under the things that can go wrong with human controllers. Mode confusion arises when the mode of the controlled system is different than the controller thinks it is. Mode confusion has contributed to many accidents and must be considered when generating causal scenarios.

In fault-handling mode, the automated controller may change its behavior because it believes there is a fault in the controlled process or in itself and different behavior is necessary to provide safe control.

3. Supervisory Mode

This mode identifies who or what is controlling the automation at any time, when multiple supervisors may be in control of (provide control commands) to the automated controller. Basically, the

supervisory mode allows coordination of control action implementation among multiple supervisors. For example, a flight guidance system in an aircraft may be issued direct commands by a human pilot or may receive commands from other system components, which could be either human or automation. While Figure G.2 abstracts all automated controllers into one box for generality, there may be and usually is more than one automated controller. One controller may control multiple others in a hierarchical arrangement, or multiple controllers may operate in parallel, or both. This, of course, can get very complicated, e.g., the multiple controllers may each be supervised by different controllers, etc.

4. Display Mode

This model specifies what type of information should be displayed to a human controller of the automated controller. The current display mode will affect both the information provided as well as how the user will interpret it. One of the components of the model of a human controller in Figure G.2 is the current display mode he/she is seeing. If these display modes (in the human and the automation) become inconsistent, then serious problems can result, i.e., the behavior on the part of one or both can lead to a system hazard. This type of mode confusion is called "Interface Interpretation Mode Confusion" and is described further in Chapter 9 of *Engineering a Safer World*.

Model of the Human Controller

In some systems today, a model of the state of the human controller, such as drowsiness or inattention, is used by the automated control algorithm to determine its behavior. Increasingly, sensors are being used to gauge the state of human controllers. For example, cars try to detect whether the driver is intoxicated or inattentive through various types of sensors that provide input to the automated controller. A dotted line from the sensor to the human controller is shown in Figure G.2 to denote a design where the human gets feedback about whether the sensor is working or what the state of the sensed variables are (e.g., the driver's intoxication level).

Model of Other Controllers

If the automation has multiple potential controllers, then the automation may need to have a model of important state variables in those controllers.

Environmental Inputs

There may be inputs from the environment that go directly to the automated controller and not through the human controller. For example, some automation today gets direct information about location from GPS. GPS location information is calculated from signals in the environment (e.g., through antennas) and goes into the automated controller, not through the human controller. In the newest aircraft, there is something called ADS-B (Automatic Dependence Surveillance) which can be used to communicate information between different aircraft, again without going through the human controller. Note that this does not include information about the state of the aircraft itself (e.g., input from pitot tubes) which is sent via sensors in or on the controlled process and is part of the model of the controlled process.

Other controllers/systems

In some complex systems today and certainly increasingly in future systems, there may be multiple controllers of the automation. For example, there may be the usual human control of the aircraft automation with perhaps some control of the aircraft automation outside the aircraft itself (usually on the ground but it could also reside on another aircraft to which the automated aircraft is tethered), which may be human or automated. The control responsibilities may be shared by different controllers.

Figure G.2 shows only one controller (in tan) and the possibility of others in a box connected to the automation as each of the controllers will have the same internal details.

As mentioned when describing the operational mode, the automation may need to know what controller is controlling it at any time. With shared control comes the possibility of conflicting commands. Some confusion may result from inconsistencies between the models in the automation and the models in its controllers and lead to UCAs. The possible types of inconsistencies and confusion can provide important contributions to causal scenario generation.

Transmission of information between the automation and its supervisor(s)

Human controllers transmit control commands via various types of controls and get feedback from the automated controller through displays. Once again, flawed transmission of information between the automated controller and its controllers will be an important (although straightforward) part of the causal scenario generation process.

Direct changes to the automated controller that do not go through the control algorithm

In some systems today, and certainly increasingly in future systems, the control algorithm and other software, as well as all the internal models used by the automation's control algorithm may be directly changeable from the outside without the human controller knowing about it. For example, on some aircraft, new software may be updated on an aircraft over the internet without the pilot or even the airline's maintenance personnel being involved (for example, the changes may originate with an OEM). The changes may affect both the control algorithm itself and the models used by it, such as maps and other information in the automated controller and indirectly (perhaps through displays) by human controllers. There are clearly potential safety and especially security issues here. Such changes may provide the potential for generating the contextual factors in a UCA.

Human Controllers

Human controllers are the most complex component of the model in Figure 1 (although system designers are quickly increasing the complexity of automated controllers) and thus provide some of the most important parts of the causal scenario generation process. There are many relevant components of the model that will contribute, including control action generation and mental processing and the mental models related to these functions.

Control Action Generation/Mental Processing

Although these two functions could be modeled as different components, they have important interactions that can be lost in such a separation. Therefore they are combined here. This component has two basic functions: generating control actions and updating mental models.

<u>Generate control actions</u>:

Control actions are generated using information from all the models in the human controller noted in Figure 1 (environment, automation, and controlled process) as well as training, written procedures, and experience. Control action generation will be affected by commands from other controllers with which the human operator interacts. For example, a pilot is partially controlled by control actions or information received from the airline operations center or from other controllers, such as Air Traffic Control or onboard instruments and systems such as TCAS (Traffic Alert and Collision Avoidance

System).[40] When there is unclear delineation of responsibilities for control, coordination challenges increase as do the causal scenarios for unsafe control actions. For example, the FAA defines the relationship between pilots and airline operations centers as 50/50 with respect to making some specific decisions related to aircraft control. Without careful delineation of responsibilities (which may be ultimately the responsibility of the Director of Operations for the airline), confusion about who is making the decisions can lead to hazards and accidents (and this has happened).

There are, of course, a very large number of incorrect behaviors possible during control action generation. The UCA generation process (the results of which are documented in the UCA tables) to identify the context in which control actions are unsafe, along with the process used by the human controller to identify the current context, will serve to limit the number of causal scenarios generated. During the scenario generation process, the analyst will need to determine how the identified context associated with the UCA could occur and why the controller might be confused about the current context.

The human controller can in some systems interact directly with the controlled process (shown by dotted lines between the human controller and the actuators and sensors and the controlled process) without going through the automation.

Update mental models:

Besides generating control actions, a major responsibility of human controllers is to update their mental models. Mental models are always updated through processing by the human mind, although some of this processing may be subconscious. Thus there are two-way arrows in the model (humans both update and use information in their mental models and things can go wrong in both functions).

The information the human uses in this updating is partially obtained through the displays, partially through inputs from its controllers or other controllers with which it interacts (e.g., information transmitted from air traffic control or the airline operations center), and partially from direct sensory input to the controller (e.g., looking out the window and seeing a tornado approaching or feeling turbulence on an aircraft). For example, a pilot receives information (such as NOTAMs) sent from the airline operations center or from other controllers, such as Air Traffic Control. In some systems, the human operators have the ability to directly observe the actuators operating, which is depicted in Figure G.2 by a dotted line from the actuator to the human controller. For example, in some plants operators may be able to walk up to the actuators and valves and watch them open and close to check for proper operation. As another example, pilots usually do a walk-around before taking off or during a pre-flight inspection they may move the controls and directly watch the aircraft ailerons to ensure they move in the correct direction. Human controllers may also receive input from the state of the controls, for example, the driver may see the steering wheel move on its own and assume that the automation has issued a control command related to steering.

Model updating may also be affected by the training that the human operator (controller) receives. Another subtle type of input may come from the reaction of the controls when the pilots operate them (shown as a dotted line in Figure G.2 from the controls to the human controller). Operators may make assumptions about the state of the automated controller and controlled process based on the

[40] While TCAS could be considered to be an automated controller (in the middle of the hierarchical control structure in Figure G.2), pilots are required to follow TCAS resolution advisories (control commands) and thus in some respects TCAS controls the pilots. However, pilots can input settings for TCAS to control how it generates resolution advisories. As more and more functions are automated on aircraft and other systems, various types of "partnerships" between humans and automation will be necessary and coordination challenges will increase.

movement of controls, such as the movement of the control column on an aircraft or the steering wheel on an automated car.

Interactions between updating models and generating control actions:

As with automated controllers and their updating of their process models, there can be important interactions between the control generation process and the mental models of human controllers similar to those that were described for automated controllers. For example, if a human controller issues a command to "Do X," human controllers are likely to assume that the "Do X" command has been executed by the Automated Controller and the actuators for the Controlled Process and unconsciously update their mental models about the state of the automation and/or the controlled process. If, in reality, the "Do X" command is never executed, then the process models of the human controller will be inconsistent with the actual state of the automation and/or controlled process. There can be many reasons why the command may not be executed, for example, it may be blocked and ignored if the automation considers the command to be unsafe or not possible at that point in time or there can be faults or failures in the controlled process.

This same type of flawed updating of process models was described in the section on the automated control algorithm. But in the case of the automation, the algorithm can be checked to ensure that this problem does not occur. It is more difficult to preclude this from human mental processing.

Feedback about the non-execution of the command may be provided to the human controller, but the human may not notice it because of distraction or other reasons.

Mental Models used by Human Controllers

Four types of mental models are identified in Figure G.2 for the human controller: a model of the environment, a model of the state of the automation, a model of the controlled process, and a model of other controllers. Any of these models may be inconsistent with the actual state and thus lead to unsafe control actions. These models have similar content and use as the same models in the automated controller, although the reasons for their being flawed may be quite different. The causal scenarios must include potential reasons for (causes of) dangerous inconsistencies that can give rise to the contextual conditions in the UCAs.

Because the human controller is usually controlling both the automation (directly) and the controlled process (indirectly but in some cases directly), humans need models of the states of both.

In order to supervise automation, humans need to have basic understanding of how the automation operates. Confusion about this can lead to unsafe control and losses.

There are a large number of reasons why human mental models may be incorrect. Some of the same reasons given earlier for models being incorrect in automated controllers may apply. But human processing presents many other opportunities. We highly recommend that human factors experts be part of the causal scenario generation involving human controllers. Here is a list of some (but only some) factors that should be considered:

- Mode confusion is a common cause of accidents in mode-rich systems where the human is confused about the mode of the automation or the automation may be confused about the current mode of the controlled process. Mode confusion may be caused by incorrect updating in process models, inputs may be incorrect or delayed, updating may be delayed or the human controller may be informed of the mode change but does not notice or process that information. When automation can change the mode of the controlled process without being directed to do so by the human controller, mode confusion and potential unsafe control actions can result.
- "Situation awareness" is commonly cited as the cause of accidents without a careful definition of what that term means. A large number of errors can fall into this category. Note that the

modeling used in STPA can identify situation awareness errors as they simply mean that the human controller mental models do not match the real state.

- Humans can easily be confused by automation that is nondeterministic or acts one way most of the time but in a few special cases behaves differently. Such automation also greatly increases training difficulties.
- Some automation is programmed such that the logic can result in side effects that are not intended or known by human controllers.
- While the design of the system may include feedback to humans, there are many reasons why that feedback may not be noticed such as distraction or lowered alertness when monitoring something that rarely changes or is rarely incorrect.
- Complacency and overreliance on automation by humans is increasingly becoming a problem in automated systems today.
- Automation may fail so gracefully that the human controller does not notice the problem until late in the process. Humans may also think that automation is shut down or has failed when it has not. This type of problem has arisen when robots and humans must work in the same areas. The logout/tagout problem, where humans think energy is off but is actually on, leads to a large number of accidents in the workplace.

This list is only meant to indicate that there are many causes that must be considered when humans are part of a system. It is far from exhaustive. Working with a human factors expert is recommended if causal scenarios need to get to this level of detail in order to be eliminated or controlled.

Some Final Thoughts

The model and guidance provided in this appendix should not be used as a checklist in generating causal scenarios. It is almost surely incomplete, particularly for some unique types of systems. You will need to think carefully yourself about how your system works. We have found it very helpful to have experts on the design and operation of the system participate in causal scenario generation. The model in this appendix may be helpful, however, in precluding some basic omissions and in helping to identify a more complete set of scenarios.

We also caution that trying to look at all potential flaws in the system and see if they can lead to a UCA, basically the process used in FMEA, is hopeless for complex systems. Always start with the hazards and with the UCAs and their contextual factors and then work backward from them to identify potential causes. There may still be a large number, but orders of magnitude fewer than trying to identify everything that can go wrong and then see if it will lead to a UCA.