Chapter 2
Complexity and Safety

Nancy G. Leveson

Abstract. Complexity is overwhelming the traditional approaches to preventing accidents in engineered systems and new approaches are necessary. This paper identifies the most important types of complexity related to safety and discusses what is necessary to prevent accidents in our increasingly complex engineered systems.

1 The Problem

Traditional safety engineering approaches were developed for relatively simple electro-mechanical systems. The problem is that new technology, especially software, is allowing almost unlimited complexity in the systems we are building. This complexity is creating new causes of accidents and changing the relative importance of traditional causes. While we have developed engineering techniques to deal with the older, well-understood causes, we do not have equivalent techniques to handle accident causes involving new technology and the increasing complexity of the systems we are building. A potential solution, of course, is to build the simpler systems, but usually we are unwilling to make the necessary compromises.

Complexity can be separated into complexity related to the problem itself and complexity introduced in the design of the solution of the problem. For complexity that arises from the problem being solved, reducing complexity requires reducing the goals of the system, which is something humans are often unwilling to do. Complexity can also be introduced in the design of the solution of the problem and often this "accidental complexity" (in the words of Brooks [cite]) can and should be eliminated or reduced without compromises on the basic system

Nancy G. Leveson
Aeronautics and Astronautics
Massachusetts Institute of Technology

goals. In either case, we need new, more powerful safety engineering approaches to dealing with complexity and the new causes of accidents arising from it.

2 What Is Complexity?

Complexity is subjective; it is not in the system itself but in the minds of observers or users of the system. What is complex to one person or at one point in time may not be to another. Consider the introduction of the high-pressure steam engine in the first half of the nineteenth century. While engineers quickly amassed information about thermodynamics, they did not fully understand what went on in steam boilers, resulting in frequent and disastrous explosions. Once the dynamics of steam were fully understood, more effective safety devices could be designed and explosions prevented. While steam engines may have seemed complex in the nineteenth century, they no longer would be considered complex by engineers. Complexity is relative and changes with time.

With respect to safety, the basic problem is that the behavior of complex systems cannot be thoroughly planned, understood, anticipated, and guarded against, that is there are "unknowns" in predicting system behavior. The critical factor that differentiates complex systems from other systems is *intellectual manageability*.

We can either not build and operate intellectually unmanageable systems until we have amassed the knowledge to fully understand their behavior or we can use tools to stretch our intellectual limits and to deal with the new causes of accidents arising from increased complexity.

Treating complexity as one indivisible property is not very useful in creating tools to deal with it. Some have tried to define complexity in terms of one or two properties of a system (for example, network interconnections). While useful for some problems, it is not for others. I have found the following types of complexity of greatest importance when managing safety:

- *Interactive complexity* arises in the interactions among system components.
- *Non-linear complexity* exists when cause and effect are not related in any obvious (or at least known) way.
- *Dynamic complexity* is related to understanding changes over time.
- *Decompositional complexity* is related to how we decompose or modularize our systems.

Other types of complexity can certainly be defined, but these seem to have the greatest impact on safety.

The rest of the paper discusses each of these types of complexity, their relationship to safety, and how they can be managed to increase safety in the complex systems we build.

2.1 Interactive Complexity

The simpler systems of the past could be thoroughly tested before use and any design errors identified and removed. That left only random component failure as the cause of accidents during operational use. The use of software is undermining this assumption in two ways: software is allowing us to build systems that cannot be thoroughly tested and the software itself cannot be exhaustively tested to eliminate design errors. Note that design errors are the only type of error in software: Because software is pure design, it cannot "fail" in the way that hardware does (including the hardware on which the software is executed). Basically, software is an abstraction.

One criterion for labeling a system as interactively complex, then, is that the level of interactions between the parts of the problem has reached the point where they no longer can be anticipated or thoroughly tested. An important cause of interactive complexity is coupling. Coupling leads to interdependence between parts of the problem solution by increasing the number of interfaces and thus interactions. Software has allowed us to build much more highly coupled and interactively complex systems than was feasible for pure electro-mechanical systems.

Traditionally, accidents have been considered to be caused by system component failures. There may be single or multiple failures involved, and they may not be independent. Usually some type of randomness in assumed in the failure behavior.

In interactively complex systems, in contrast, accidents may arise in the interactions among components, where none of the individual components may have failed. These *component interaction accidents* result from system design errors that are not caught before the system is fielded. Often they involve requirements errors, particularly software requirements errors. In fact, because software does not "wear out," the only types of errors that can occur are requirements errors or errors in the implementation of the requirements. In practice, the vast majority of accidents related to software have been caused by software requirements errors, i.e., the software has not "failed" but did exactly what the software implementers wanted it to do but the implications of the behavior from a system standpoint were not understood and led to unsafe system behavior.

Component interaction accidents were noticed as a growing problem starting in the Intercontinental Ballistic Missile Systems of the 1950's when interactive complexity in these systems had gotten ahead of our tools to deal with it. System engineering and System Safety were created to deal with these types of problems [Leveson, 1995]. Unfortunately, the most widely used hazard analysis techniques stem from the early 1960s and do not handle today's very different types of technology and system design.

An important implication of the distinction between component failure and component interaction accidents is that safety and reliability, particularly in complex systems, are *not* the same although they are often incorrectly equated.

Making all the components highly reliable will not prevent component interaction accidents or those arising from system design errors. In fact, sometimes they conflict and increasing one will even decrease the other, that is, increasing safety may decrease reliability and increasing component reliability may decrease system safety.

The distinction between safety and reliability is particularly important for software-intensive systems. Unsafe software behavior is usually caused by flaws in the software requirements. Either there are incomplete or wrong assumptions about the operation of the controlled system or required operation of the computer or there are unhandled controlled-system states and environmental conditions. Simply trying to get the software "correct" or to make it reliable (however one might define that attribute for a pure abstraction like software), will not make it safer if the problems stem from inadequate requirements specifications.

2.2 Non-linear Complexity

Informally, non-linear complexity occurs when cause and effect are not related in an obvious or direct way. Sometimes non-linear causal factors are called "systemic factors" in accidents, i.e., characteristics of the system or its environment that indirectly impact all or many of the system components. Examples of systemic factors are management pressure to increase productivity or reduced expenses. Another common systemic cause is the *safety culture*, which can be defined as the set of values upon which members of the organization make decisions about safety. The relationship between these systemic factors and the events preceding the accident (the "chain of events" leading to the accident) are usually indirect and non-linear and often omitted from accident reports or from proactive hazard analyses. Our accident models and techniques assume linearity, as discussed below.

Along with interactive complexity, non-linear complexity makes system behavior very difficult to predict. This lack of predictability affects not only system development but also operations. The role of operators in our systems is changing. Human operators previously were directly controlling processes and usually following predefined procedures. With the increase in automation, operators are now commonly supervising automation that controls the process rather than directly controlling the process itself. Operators have to make complex, real-time decisions, particularly during emergencies, and non-linear complexity makes it harder for the operators to successfully make such real-time decisions [Perrow, 1999]. Complexity is stretching the limits of comprehensibility and predictability of our systems.

Newer views of human factors reject the idea that operator errors are random failures [Dekker, 2005; Dekker, 2006]. All behavior is affected by the context (system) in which it occurs. Human error, therefore, is a symptom, not a cause; it is a symptom of a problem in the environment, such as the design of the equipment and human-automation interface, the design of the work procedures, management pressures, safety culture, etc. If we want to change operator

behavior, we need to change the system in which it occurs. We are designing systems in which operator error is inevitable and then blaming accidents on operators rather than designers. Operator errors stemming from complexity in the system design will not be eliminated by more training or telling the operators to be more careful.

2.3 Dynamic Complexity

Dynamic complexity is related to changes over time. Systems are not static, but we often assume they are when we design and field them. Change, particularly in human and organizational behavior, is inevitable as is change (both planned and unplanned) in the non-human system components.

Rasmussen [1997] has suggested that these changes often move the system to states of higher risk. Systems migrate toward states of high risk, according to Rasmussen, under competitive and financial pressures. The good news is that if this hypothesis is true, the types of change that will occur are potentially predictable and theoretically preventable.

We want flexibility in our systems and operating environments, but we need engineering design and operations management techniques that prevent or control dangerous changes and detect (before an accident) when they occur during operations.

2.4 Decompositional Complexity

Interactive, non-linear, and dynamic complexity are related to the problem being solved and not necessarily the solution, although they impact and are reflected in the design of the system. For the most part, complexity in the design of the solution is not very relevant for safety. But design complexity does have a major impact on our ability to analyze the safety of a system. The aspect of design that most affects safety, in my experience, is decompositional complexity.

Decompositional complexity arises when the structural decomposition of the system is not consistent with the functional decomposition. Decompositional complexity makes it harder for designers and maintainers to predict and understand system behavior. Safety is related to the functional behavior of the system and its components: It is not a function of the system structure or architecture. Decompositional complexity makes it harder for humans to understand and find functional design errors (versus structural flaws). For safety, it also greatly increases the difficulty for humans to examine the system design and determine whether the system will behave safely. Most accidents (beyond simple causes such as cuts on sharp edges or physical objects falling on people) occur as a result of some system behavior, i.e., the system has to *do* something to cause an accident.

Because verifying safety requires understanding the system's functional behavior, designing to enhance such verification is necessary. For large systems, this verification may be feasible only if the system is designed using functional decomposition, for example, isolating and modularizing potentially unsafe functionality. Spreading functionality that can affect safety throughout the entire system design makes safety verification infeasible. I know of no effective way to verify the safety of most object-oriented system designs at a reasonable cost.

3 Managing Complexity in Safety Engineering

To engineer for safety in systems exhibiting interactive, non-linear, and dynamic complexity, we will need to extend our standard safety engineering approaches. The most important step is probably the most difficult for people to implement and that is to limit the complexity in the systems we build and to practice restraint in defining the requirements for our systems. At the least, extra unnecessary complexity should not be added in design and designs must be reviewable and analyzable for safety.

Given that most people will be unwilling to go back to the simpler systems of the past, any practical solution must include providing tools to stretch the basic human intellectual limits in understanding complexity. For safety, these tools need to be built on top of a model of accident causality that encompasses the complexities of the systems we are building.

3.1 STAMP: A New Accident Model

Our current safety engineering techniques assume accidents are caused by component failures and do not assist in preventing component interaction accidents. The most common accident causality model explains accidents in terms of multiple events, sequenced as a forward chain over time. The relationships among the events are assumed to be simple and direct. The events almost always involve component failure, human error, or energy-related events (e.g., an explosion).

This chain-of-events model forms the basis for most safety engineering and reliability engineering analysis (for example, fault tree analysis, probabilistic risk analysis, failure modes and effects analysis, events trees, etc.) and design for safety (e.g., redundancy, overdesign, safety margins).

This standard causality model and the tools and techniques built on it do not apply to the types of complexity described earlier. It assumes direct linearity between events and ignores common causes of failure events, it does not include component interaction accidents where no components may have failed, and it does not handle dynamic complexity and migration toward states of high risk. It also greatly oversimplifies human error by assuming it involves random failures or

"slips," that are unrelated to the context in which the error occurs, and that operators are simply blindly following procedures and not making cognitively complex decisions. In fact, human error is better modeled as a feedback loop than a "failure" in a simple chain of events.

STAMP (System-Theoretic Accident Model and Processes) was created to include the causes of accidents arising from these types of complexity. STAMP is based on systems theory rather than reliability theory and treats accidents as a control problem rather than a failure problem. The basic paradigm change is to switch from a focus of "prevent failures" to one of "enforce safety constraints on system behavior." The new focus includes the old one but also includes accident causes not recognized in the old models.

In STAMP, safety is treated as an emergent property that arises when the system components interact with each other within a larger environment. There is a set of constraints related to the behavior of the system components—physical, human, and social—that enforces the emergent safety property. Accidents occur when system interactions violate those constraints.

In this model of causation, accidents are not simply an event or chain of events but involve a complex, dynamic process. Dynamic behavior of the system is also included: most accidents are assumed to arise from a slow migration of the entire system toward a state of high risk. Often this migration is not noticed until after an accident has occurred. Instead we need to control and detect this migration.

The standard chain-of-failure-events model is included in this broader control model. Component failures are simply a subset of causes of an accident that need to be controlled. STAMP more broadly defines safety as a dynamic control problem rather than a component failure problem. For example, the O-ring in the Challenger Space Shuttle did fail, but the problem was that the failure caused the O-ring not to be able to *control* the propellant gas release by sealing a gap in the field joint of the Space Shuttle. The software did not adequately *control* the descent speed of the Mars Polar Lander. The public health system did not adequately *control* the contamination of the milk supply with melamine in a recent set of losses. Our financial system did not adequately *control* the use of financial instruments in our recent financial meltdown.

Constraints are enforced by socio-technical safety control structures. Figure 1 shows an example of such a control structure. There are two hierarchical structures shown in Figure 1: development and operations. They are separated because safety is usually controlled very differently in each. A third control structure might also be included which involves emergency response when an accident does occur so that losses are minimized.

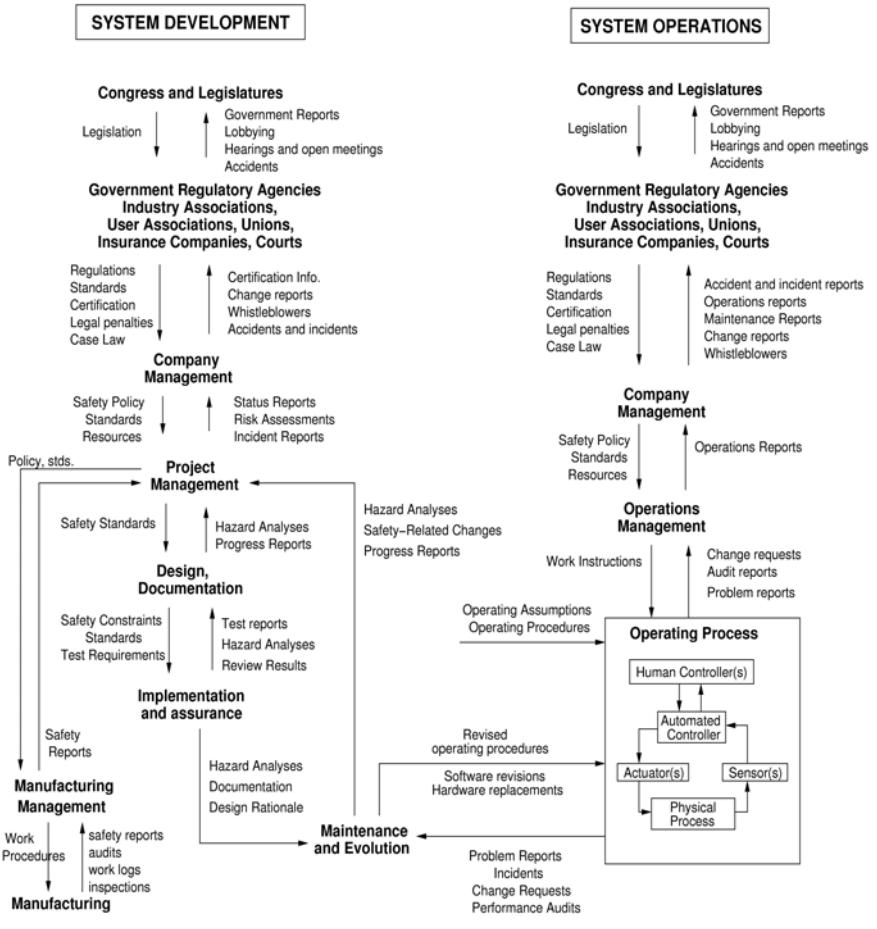

Fig. 1 An Example Socio-Technical Safety Control Structure

While Figure 1 focuses on the social and managerial aspects of the problem, the physical process itself can be treated as a control system in the standard engineering way. Figure 2 shows a sample control structure for an automobile adaptive cruise control system.

Each component in the safety control structure has assigned responsibilities, authority, and accountability for enforcing specific safety constraints. The components also have various types of controls that can be used to enforce the constraints. Each component's behavior, in turn, is influenced both by the context (environment) in which the controller is operating and by the controller's knowledge about the current state of the process.

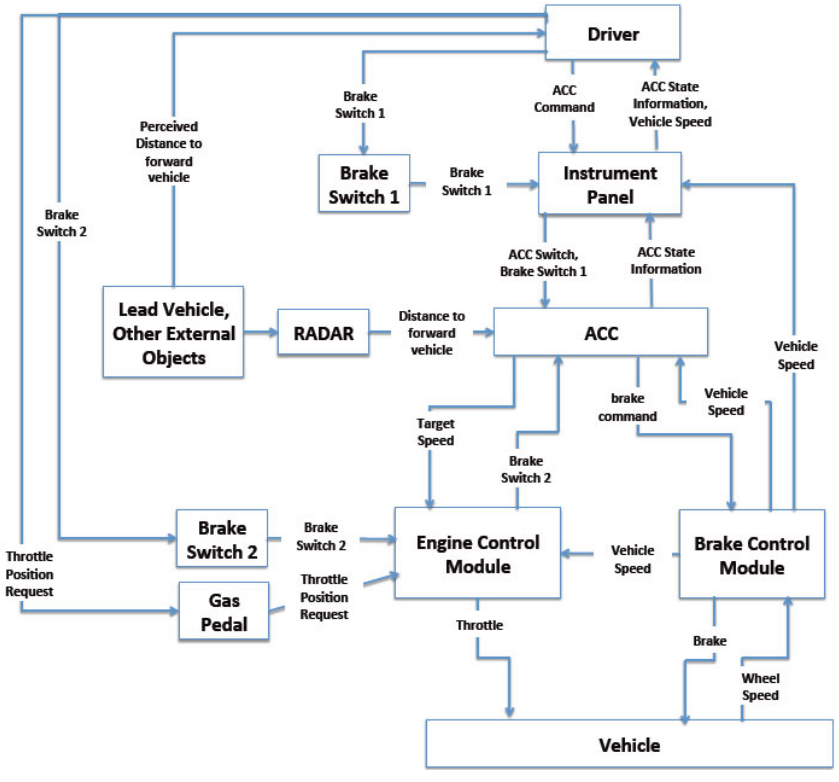

Fig. 2 A Sample Control Structure for an Automobile Adaptive Cruise Control System [Qi Hommes and Arjun Srinath]

Any controller needs to have a model of the process it is controlling in order to provide appropriate and effective control actions. That process model is in turn updated by feedback to the controller.

Accidents often occur when the model of the process is inconsistent with the real state of the process and the controller provides unsafe control actions (Figure 3). For example, the spacecraft software thinks that the spacecraft has reached the planet surface and prematurely turns on the descent engines. Accidents occur when the process models do not match the process and

- Commands required for safety (to enforce the safety constraints) are not provided or are not followed;
- Unsafe commands are given that cause an accident;
- Correct and safe commands are provided but at the wrong time (too early, too late) or in the wrong sequence
- A required control action is stopped too soon or applied too long.

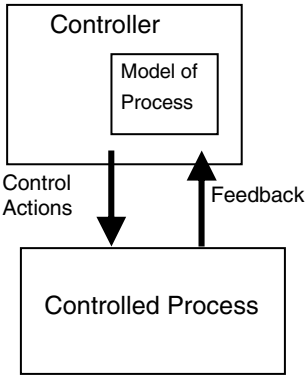

Fig. 3 Every controller contains a model of the controlled process it is controlling

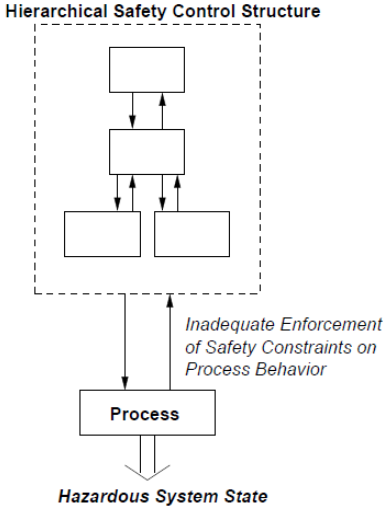

Fig. 4 In STAMP, accidents occur due to inadequate enforcement of safety constraints on system process behavior

The STAMP model of causality does a much better job of explaining accidents caused by software errors, human errors, component interactions, etc. than does a simple failure model. Figure 4 shows the overall concept behind STAMP. There are, of course, many more details. These can be found in [Leveson, 2011].

3.2 Using STAMP in Complex Systems

Just as tools like fault tree analysis have been constructed on the foundation of the chain-of-failure events model, tools and procedures can be constructed on the foundation of STAMP. Because STAMP includes more causes of accidents (but also includes standard component failure accidents), such tools provide a theoretically more powerful way to In particular, we will need more powerful tools in the form of more comprehensive accident/incident investigation and causal analysis, hazard analysis techniques that work on highly complex systems, procedures to integrate safety into the system engineering process and design safety into the system from the beginning rather than trying to add it on at the end, organizational and cultural risk analysis (including defining safety metrics and leading indicators of increasing risk), and tools to improve operational and management control of safety. Such tools have been developed and used successfully on enormously complex systems. Figure 5 shows the components of an overall safety process based on STAMP.

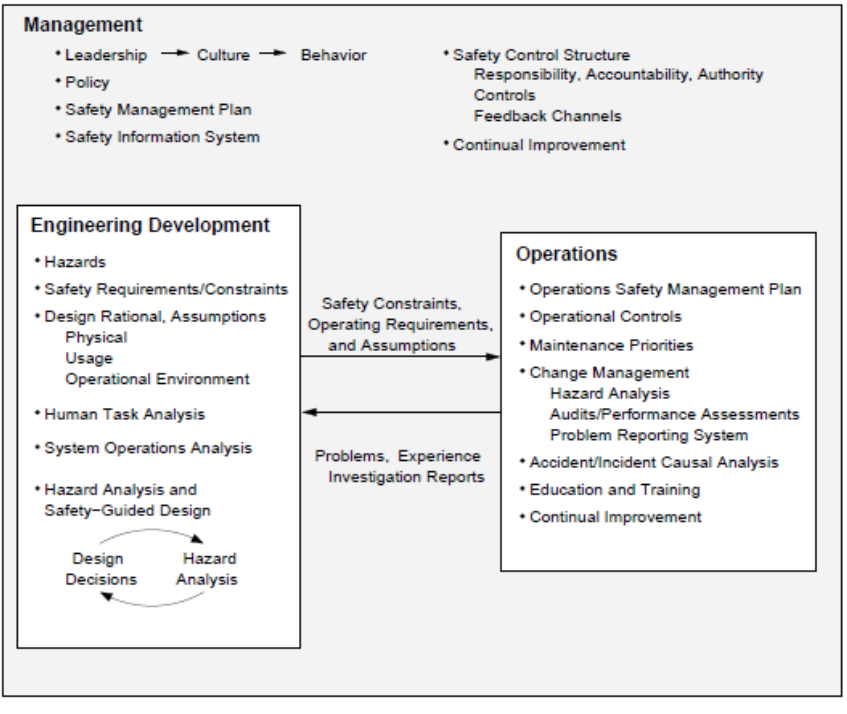

Fig. 5 The Overall Safety Process as Defined [Leveson, 2011].

4 Summary

This paper has described types of complexity affecting safety in our modern, high-tech systems and argued that a new model of accident causality is needed to handle this complexity. One important question, of course, is whether this new model and the tools built on it really work. We have been applying it to a large number of very large and complex systems in the past ten years (aerospace, medical, transportation, food safety, etc.) and have been surprised by the how well the tools worked. In some cases, standard hazard analysis techniques were applied in parallel (by people other than us) and the new tools proved to be more effective [see for example, JAXA [Arnold, 2009; Ishimatsu, 2010; Nelson, 2008; Pereira, 2006].

One lesson we have learned is the need to take a system engineering view of safety rather than the current component reliability view when building complex systems. The entire socio-technique system must be considered, including the safety culture and organizational structure. Another lesson is that safety must be built into a complex system; it cannot be added to a completed design without enormous (and usually impractical) cost and effort and with diminished effectiveness. To support this system engineering process, new specification techniques must be developed that support human review of requirements and safety analysis during development and the reanalysis of safety after changes occur during operations. changes.

Finally, we also need a more realistic handling of human errors and human decision making and to include the behavioral dynamics of the system and changes over time into our engineering and operational practices. We need to understand why controls migrate toward ineffectiveness over time and to manage this drift.

References

Arnold, R.: A Qualitative Comparative Analysis of STAMP and SOAM in ATM Occurrence Investigation, Master's Thesis, Lund University, Sweden (June 2009)

Dekker, S.: The field guide to understanding human error. Ashgate Publishing, Ltd. (2006)

Dekker, S.: Ten Questions About Human Error: A New View of Human Factors and System Safety. Lawrence Erlbaum Associate Inc., Mahwah (2005)

Ishimatsu, T., Leveson, N., Thomas, J., Katahira, M., Miyamoto, Y., Nakao, H.: Modeling and Hazard Analysis using STPA. In: Proceedings of the International Association for the Advancement of Space Safety Conference, Huntsville, Alabama (May 2010)

Leveson, N.: Engineering a Safer World: Systems Thinking Applied to Safety. MIT Press (December 2011), downloadable from `http://sunnyday.mit.edu/safer-world/safer-world.pdf`

Leveson, N.G.: Safeware: System Safety and Computers. Addison-Wesley Publishers, Reading (1995)

Nelson, P.S.: A STAMP Analysis of the LEX Comair 5191 Accident. Master's thesis, Lund University, Sweden (June 2008)

Pereira, S., Lee, G., Howard, J.: A System-Theoretic Hazard Analysis Methodology for a Non-advocate Safety Assessment of the Ballistic Missile Defense System. In: Proceedings of the 2006 AIAA Missile Sciences Conference, Monterey, CA (November 2006)

Perrow, C.: Normal Accidents: Living with High-Risk Technology. Princeton University Press (1999)

Rasmussen, J.: Risk management in a dynamic society: A modeling problem. Safety Science 27(2/3), 183–213 (1997)