White Paper on How to Perform Hazard Analysis on a "System-of-Systems"

Prof. Nancy G. Leveson
Aeronautics and Astronautics Dept.
MIT

Abstract: Many people seem to believe that a different type of hazard analysis is required for a "system-of-systems" than for just a complex system. The task to do a system-of-systems hazard analysis has even been added to the official requirements for system safety of military systems in the U.S. (MIL-STD-882) in additional to a task for traditional system hazard analysis. The definition of a system, however, as defined in System Theory, already incorporates all the supposedly new properties.

Perhaps the new term has been invented because most of traditional hazard analysis techniques do not scale up to the complexity of modern defense and other systems (essentially what is being called a system-of-systems). Much of the confusion arises from the formal definition of a system. This paper clarifies the definition of a system and shows how STPA can be used for what has been labeled (erroneously) as a system-of-system, without any changes to STPA. A very complex military system, created by composing both existing and new systems, is used as the example.

Introduction

For some reason, the term "system-of-systems" has become popular although theoretically there is no such thing. But the problem creating the impetus for a new term does exist, that is, how to handle today's increasingly complex systems using the standard hazard analysis techniques. Unfortunately, these techniques do not work for today's complex, software-intensive systems and will not scale up to handle them. Safety engineers go through the motions, usually not realizing that their results are, at the least, very incomplete. Lots of paper is produced, lots of numbers posited as reflecting risk, and none of these have much to do with reality. There is, of course, no way to know that the results are incomplete unless or until there is a mishap. Surprisingly, people rarely go back and question why the real-world results do not match the predictions after a mishap or even multiple mishaps.

In this white paper, I will define the term "system" and explain why a system-of-systems" is not a useful concept. Then I continue to show how to handle so-called systems-of-systems using STPA, that is, how to solve the problem that the term was invented to describe.

What is a System?

Theoretically, a *system* is defined as a set of components that act together as a whole to achieve some common goal, objective, or end. The components are all interrelated and are either directly or indirectly related to each other. So, a chemical plant, an airplane, an automobile, transportation in general, county government, and a television set are examples of systems. They all consist of a set of components working together to achieve a common goal. An example of a non-system is a set consisting of a shoe, an apple, and a wrench; this is not a system unless you can describe some common goal that they achieve. A purpose is basic to the concept.

There are two assumptions underlying the concept of a system: (1) the system goals can be defined and (2) the system is atomistic, that is, capable of being separated into component entities such that their interactive behavior can be described. Without being able to divide the system into separate

components, we can only talk about the thing as a whole, which limits the usefulness of the concept of a system. The interactive behavior of the system components plays a critical role in the achievement of the system objectives.

This brings us to one of the most important aspects of a system that you need to understand: A system is an *abstraction* (model) *conceived by the observer*. For the same man-made system, an observer may see a different purpose or purposes than the designers or other observers and may also focus on different relevant properties, even different goals. I view a particular transportation system as a way to take me to work while someone else may see it as a way to increase business in the area by transporting customers and employees while someone else may see a system that is polluting the area. In summary, for the same man-made system, an observers or users may see different purposes and each may see a different purpose than the designers. They will also, likely, focus on different relevant properties of the system.

For natural (not man-made) systems, we *only* have observers (and cannot question the designer) so different people may conceive of these systems in different ways. In engineering, we are almost always concerned with man-made systems. Note that social systems are usually man made.

This concept of a system as an abstraction explains the importance of system specifications. They ensure consistency of mental models among those designing, using, or viewing a system and therefore are critical for effective communication. The common specification required must define the following basic aspects of the system:

- System boundary: what is inside and what is outside (not in) the system
- Inputs and outputs: most interesting systems get inputs from the environment and send outputs to the environment.
- Components
- Structure
- Relevant interactions among components (the behavior of the components and their effect on the overall system state)
- Purpose or goals of the system that makes it reasonable to consider it to be a coherent entity[1]

Note that there may be different objectives for different people viewing the same components, which makes them a different system for each viewer.

Consider an airport. 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. While the physical aspects of an airport are the same for everyone, remember that a system is an abstraction imposed on a physical thing. Therefore, when talking about airports as a system, we need to specify the purpose of the "system" being considered at that time. Again, the term "system" is an abstraction imposed on some components by those considering that system. While the components may exist in reality, the system itself only exists in the minds of the viewers.

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. System views may be created by varying groups when considering an existing system.

---

[1] Peter Checkland, *Systems Thinking, Systems Practice*, New York: John Wiley & Sons, 1981.

With this basic understanding of a system as an abstraction, we can now consider more aspects about this abstraction we call a "system." In fact, there exists a formal, scientific foundation for dealing with systems. General Systems Theory was introduced 60-70 years ago. It is surprising that although engineers and others use the term "system" freely, they seldom have learned the theoretical underpinnings.

One important theoretical aspect is the concept of a *subsystem*. Consider our airport "system" again. Different components of the airport may be included in a particular viewer's "airport" system. The airline view of an airport may include passenger check-in counters, ramps to the planes, and taxiways. A commercial view may include only shops and customers. Notice 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. 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. In the airport example, different abstractions for an airport may be laid upon the physical components by different users and for different uses.

A second important understanding about systems is that the definition of a system is *recursive*, that is, the definition is made in terms of itself. Systems themselves may be part of a larger system or be divisible into subsystems (viewed as components of the overall system). Figure 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. Note that this is not a connection diagram. It is multiple (in this case two) views of the same system at different levels of detail.

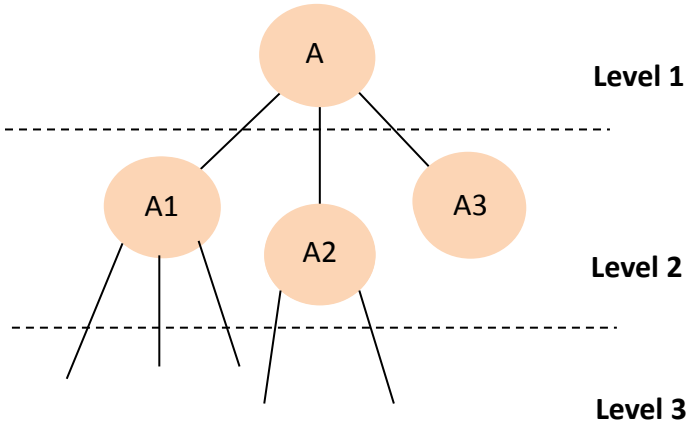


Figure 1: The abstraction System A made be viewed as composed of three subsystems, where each subsystem is itself a system.

When viewed as part of System A, then A2, for example, is a subsystem of A. But A2 can be viewed as a system by itself. Figure 2 shows an abstraction where System A is conceived as part of a larger system AB. Each of these abstractions may provide alternative views of a system, each useful for its own purposes.

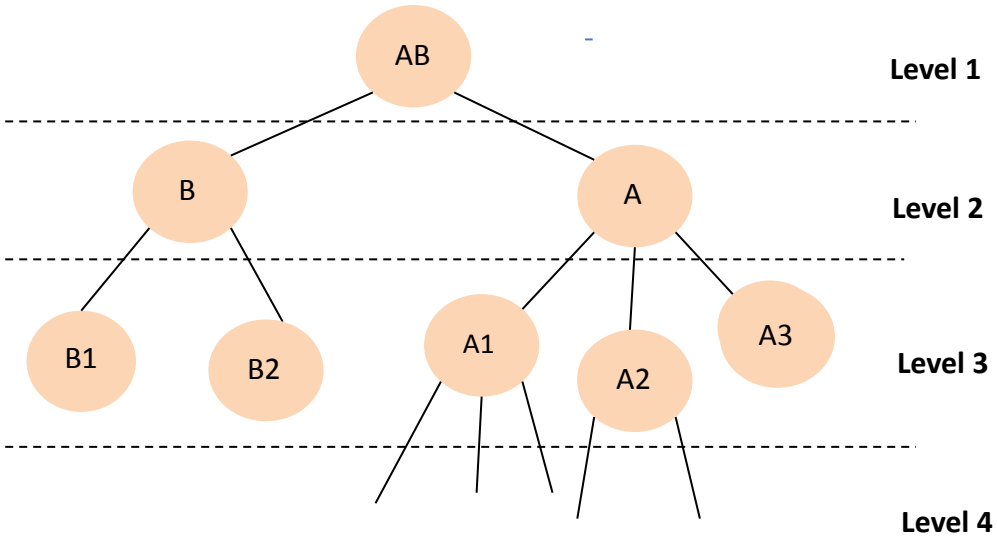


Figure 2: System A can be viewed as a component (subsystem) of a larger system AB

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" (with subsystems, which can be considered to be systems themselves) using the definition of a system as defined in system theory and as the concept is used in engineering. More about this in the next section of this paper.

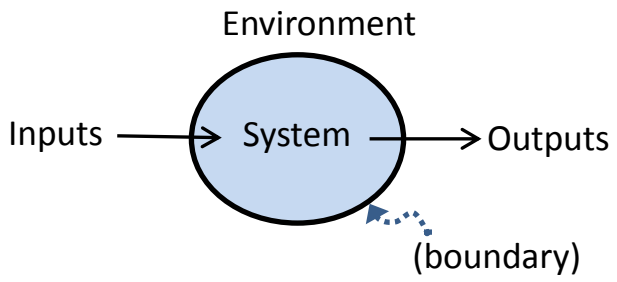
Systems have *states*. A state is a set of relevant properties describing the system at any point in time. Some properties of the state of an airport viewed as an air transportation system may be the number of passengers at a particular gate, where the aircraft are located and what they are doing (loading passengers, taxiing, taking off, or landing). In the commercial airport system, the state may include the number and type of stores, whether they are open or not, how many customers each has, etc. Some variables (and thus states) will change frequently (and thence the state will change frequently), such as the number of customers in the stores, while other variables may change much less frequently, such as the number and type of stores or commercial enterprises.

The concept of a system state is important as a hazard (informally) is a particular system state that can lead to a loss. Examples are an aircraft that is too close to a mountain or a chemical plant where the pressure in a tank exceeds a threshold value. Events lead to state changes. We often loosely talk about hazards as events (release of a chemical from a plant, for example), but that can be translated into the equivalent state: for example, chemicals are in the air within and outside the plant boundaries. We could be more exact and demanding, but it sometimes is more natural to talk about events rather than states if there is not a formal mathematical process being described.

The components of the state that are relevant depend on how the boundaries are drawn between the system and its environment. Because the goal of safety engineering is to eliminate or control hazards, the engineer needs to define hazards as states within their system design space, i.e., states they have control over.

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 and the person creating the abstraction.



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.

One of the most important properties of systems is that they have "emergent" properties. Historically, scientists and engineers have dealt with complex systems by using decomposition. Decomposition separates systems into atomistic components, analyzes each component for a property in isolation from the rest, and then gets a system value by combining the values from the individual components. This may work for some important properties. Consider, for example, weight. If we weigh each of the components of a system, combining them will provide a weight for the system as a whole. Weight is a decomposable (non-emergent) property.

The usefulness of decomposition relies on an assumption that separating the system into parts does not distort the phenomenon you are interested in. That is, each component or subsystem operates independently; components act the same when they are examined individually as when they are playing their part in the whole; and the interactions are all direct and therefore the results of analyzing each can be examined through only looking at the direct channels between them. If the components have indirect interactions and complex interactions such as feedback loops, then the simple summation of properties for the individual components will not provide an accurate accounting of the property for the system as a whole.

The reason for modern systems theory is that most interesting systems have properties for which these assumptions are not true and therefore properties for which analytical decomposition does not provide useful answers. These are called "emergent" properties. Emergent properties are properties that are not in the individual components but "emerge" when the components operate together. Most interesting system properties are emergent and, in fact, it was to deal with these emergent properties that systems theory was created.

Before the middle of the last century, most systems were simple enough that analytic decomposition worked adequately most of the time. After WW II and with the advent of digital computers, system complexity increased so much that analytic decomposition was no longer useful. Systems theory was created to deal with these increasingly complex systems.

Emergent properties are so important, let's look at them a little more. To deal with emergent properties, systems theory (and methodologies based on systems theory) focus on a system taken as a whole, not on the parts separately. Emergent properties can only be treated adequately in their entirety by taking into account all social and technical aspects. That is, systems theory works on complex sociotechnical systems.

Emergent properties arise from the relationships among the parts of the system, that is, how the parts interact and fit together. You have probably heard the adage "The whole is greater than the sum of the parts." This is the basic concept in systems theory and provides a simple explanation of emergence.

Safety and security are emergent properties. Looking only at a valve, for example, it is not possible to determine whether an aircraft or an automobile that uses that valve in its design will be safe. Safety, like other emergent properties, depends on how the components operate and interact as a whole. In fact, most interesting system properties are emergent, such as quality, efficiency, throughput, etc.

Consider the property "throughput" and the National Airspace System. If each airline optimized the routes of each of its planes without considering the routes of other airlines, the airspace system would end up in chaos, with planes from different airlines interfering with each other and all trying to land or take off at the same time. The attempt to optimize individual component behavior would simply lead to nobody achieving their goals and the system throughput being diminished. That's why we have an air traffic control system. It coordinates the routes of all the planes so that the system throughput is optimized, if not the throughput of each airline. In the end, the *average* throughput for all the airlines is optimized.

It's surprising that most of the difficult problems today that people lament have no solutions are actually solved by something that was invented over 60 years ago. Unfortunately, systems theory has never been taught extensively in most universities nor has much of system engineering actually adopted it. It has had much more impact on other fields, such as biology. In the early days of system engineering and system safety[2], systems theory was the basis for the development of techniques and tools. But over time, people went back to what was traditional and familiar, namely, analytic decomposition.

Now that we have established the definition of a system and the basic concepts in system theory, we can turn to the frequently used term "system-of-systems."

Why a System-of-Systems is not a Useful Technical Term

People can, of course, introduce and use any terminology they want. But a proliferation of terminology can be confusing and misleading, as in this case.

As the concept of a "system of systems" does not exist in systems theory, I have tried to figure out how people are using this term. For the most part, it seems to have replaced the term "system" and not added anything new. Wikipedia defines it as "A **System of systems** is a collection of task-oriented or dedicated **systems** that pool their resources and capabilities together to create a new, more complex **system** which offers more functionality and performance than simply the sum of the constituent **systems."** Elsewhere, it is defined as "System of systems are large-scale concurrent and distributed systems the components of which are complex systems themselves." Both of these definitions are identical to the definition of a system, with "system" replacing the term "system of systems" and subsystem replacing "system."

Wikipedia also suggests "While the individual systems constituting a system of systems can be very different and operate independently, their interactions typically expose and deliver important emergent properties." The concept of emergence is basic to the definition of a system in systems theory and is not new. Again, this is all consistent with the definition of a system as defined in system theory.

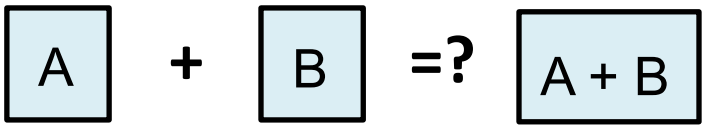
---

[2] If you go back to the foundational papers for system engineering and system safety written in the 1950s and 1960s, you will find that system theory is clearly stated as the basis. Understanding why these concepts were lost over time would require a historical analysis.

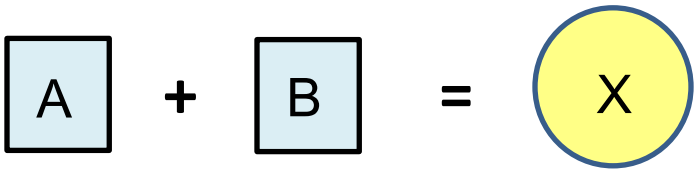
Lots of other supposed distinctions have been made between systems and systems-of-systems, all of which are already included in the definition of a system. For example, some suggest that the difference is that some of the subsystems already exist in a system of systems and are put together. There is no implication in the definition of a system that everything must be new. In fact, this is rarely the case. Nobody ever builds a completely new system without using components (e.g., screws or even radars) that are all newly designed. Again, the definition is the same. The fact that parts of the system already exist does not make any difference in creating an abstraction that combines components into something the user wants to call a system. Almost always such components already exist.

Other supposed distinctions are either true of all real systems or are irrelevant to the basic definition. From all of this, I surmise that the problem people are trying to handle is "complexity" and the growing complexity of the systems we are attempting to build, not new concepts. I understand their frustration with analytic decomposition, but new terms need not be created when a scientific field exists that already covers their concerns.

Finally, I have occasionally heard people describe a system of systems as one that joins two or more existing systems. The abstraction concept of a system already includes this as simply a subcase. But I also suspect that there is a subtle return to thinking in terms of decompositional approaches here and an implication that the properties of the two systems can be combined, i.e., "summed," to get a system of systems analysis. Unfortunately, basic systems theory shows us this is incorrect. The assumption seems to be that:



But this assumption is untrue. In systems theory, when you put two systems together you get a new and totally different system with *different emergent properties*.



In the new abstraction, it is irrelevant whether the components that are combined existed or not before. The critical concept here is that the emergent properties of the combined new system are different and cannot simply be handled by combining the evaluation of the emergent properties (such as safety or risk) identified for the individual systems.

In summary, the definition proposed for a "system of systems" all fit the original and formal definition of a "system." Therefore, nothing new is added by introducing a redundant term and, in fact, confusion is a common result. Suggesting that something new is involved is hindering progress.

So, all this is nice, you're thinking, but how does it help me with my problems? The final section of the paper shows how to analyze a complex system or system-of-systems or whatever you want to call it using STPA. An example is provided of the successful hazard analysis of a real system that is one of the most complex engineered systems ever created.

How to Analyze a System or "System-of-Systems" using STPA

This section assumes that the reader is familiar with STPA. If not, first looking at the STPA Handbook would be helpful. The example used is a real defense system, but details need to be omitted for obvious reasons. It clearly fits the definition of a system and satisfies every property that supposed systems-of-systems are defined as having. It was, in fact, one of the first complex industrial systems to be analyzed using STPA back in 2005.[3]

The U.S. Ballistic Missile Defense System (BMDS) is a layered defense system to defeat all ranges of threats in all phases of flight (boost, midcourse, and terminal). The BMDS integrates into a single system a number of systems that have been developed independently, including sea-based sensors on the Aegis platform, upgraded early warning radars (UEWR), the Cobra Dane Upgrade (CDU), Ground-based Midcourse Defense (GMD) Fire Control and Communications (DFC/C), a Command and Control Battle Management and Communications system (C2BMC), and Ground-Based Interceptors (GBI). Future upgrades were originally planned to introduce additional systems into the BMDS, including Airborne Laser (ABL) and Terminal High Altitude Area Defense (THAAD). Some of the components of BMDS existed previously, some were upgrades of existing systems (such as UEWR and CDU), and some were new. In BMDS, existing systems were being used in a different environment than that for which they were originally developed.

All the systems being integrated into BMDS had their own active safety programs, but considerable complexity, coupling, and risk was introduced by integrating them into a single system. Those who performed the BMDS hazard analysis and selected STPA stated that the effort required a hazard analysis methodology that (1) considers hazards and causes due to complex system interactions (more than just failure events); (2) provides guidance in conducting the analysis, (3) comprehensively addresses the whole system, including hardware, software, operators, procedures, maintenance, and continuing development activities; and (4) focuses resources on the areas of the system with the greatest impact on safety and risk.[3]

The effort to apply STPA began late in development, prior to deployment and field testing. This timing limited the flexibility for design changes in response to the findings and made requiired changes much more expensive. Ideally, STPA should have been started early in the BMDS concept development stage to provide the most benefit and least additional costs. The late use led to extra difficulties that required performing hazard analyses where existing analysis work was missing or inadequate; working with existing engineering artifacts (including documentation) that was erroneous, ambiguous, incomplete, or outdated; and making recommendations for hazard mitigations late in the system life cycle with resulting extra costs and limited flexibility for changes.

The BMDS hazard analysis (which was focused on inadvertent launch) involved two people working for 5 months. Neither was originally familiar with the system (or the subsystems) and much of the time was spent learning about how the system worked.

For obvious reasons, details about the real BMDS cannot be provided here. Instead, as in the paper written by the people performing the analysis, a Fictional Missile Intercept System (FMIS) is used as an example. The FMIS, like the BMDS, uses a hit-to-kill interceptor that destroys incoming ballistic missiles through force of impact. Examples of the analysis on the FMIS are shown here and the overall results from the real analysis on the BMDS are summarized at the end.

Hazard: *The FMIS inadvertently launches an interceptor missile.*

---

[3] Steven J. Pereira, Grady Lee, and Jeffrey Howard, A System-Theoretic Hazard Analysis Methodology for a Non-Advocate Safety Assessment of the Ballistic Missile Defense System, *Proceedings of the 2006 AIAA Missile Sciences Conference*, Monterey California, 14-16 November, 2006.

This hazard traces to a requirement in the top-level system specification to eliminate inadvertent launch.

Figure 3 shows the high-level FMIS operational control structure. The development structure is not shown. This operational control structure must enforce the system safety requirements and constraints during the life time of the system. The goal of the analysis is to identify any potential inadequate enforcement of the constraints that might arise from the system design.

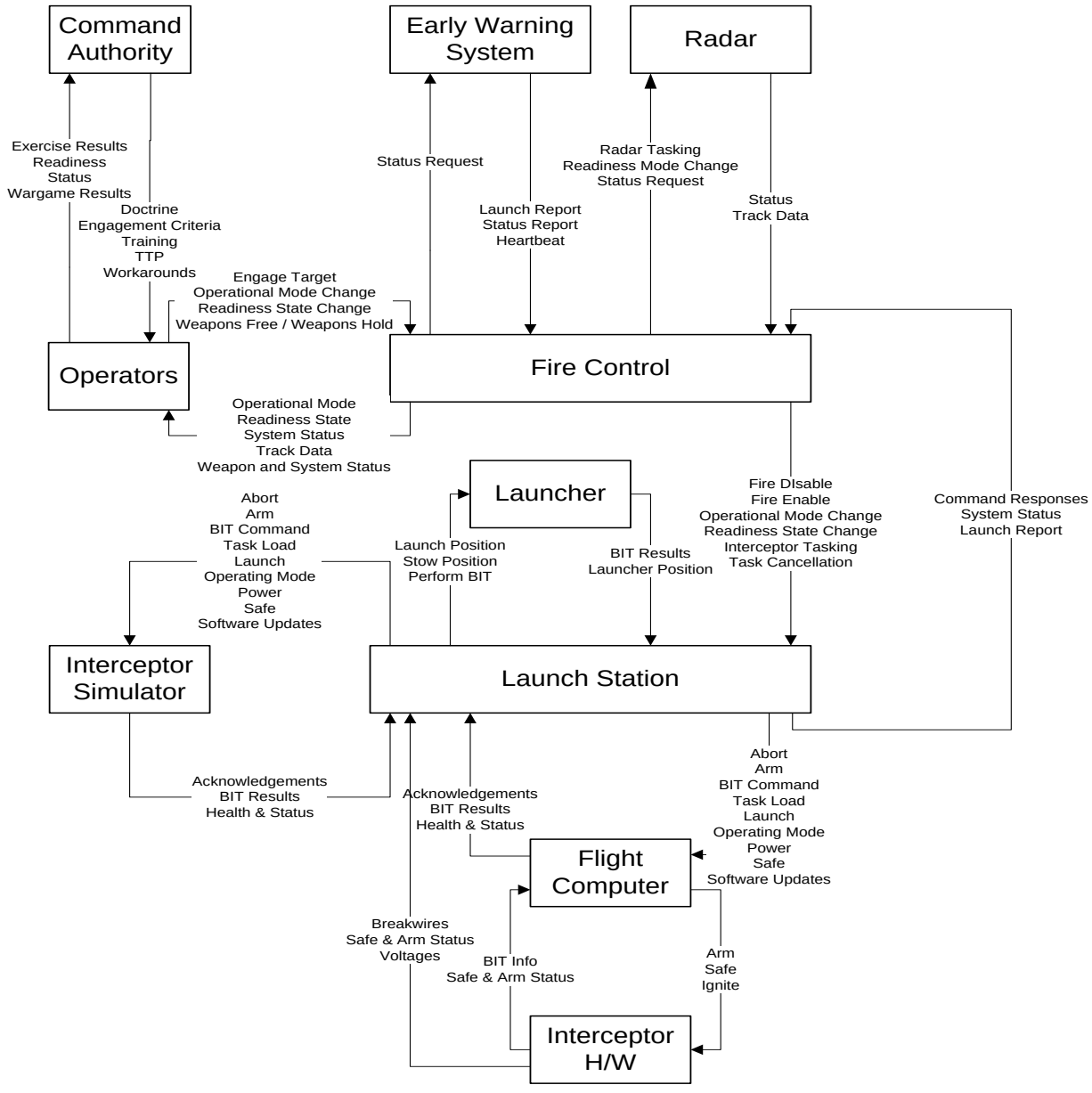


Figure 3: The High-Level FMIS control structure

STPA starts with a very high-level view of the entire system. Note that the control structure diagram is not a data flow diagram and does not represent message names from interface specifications—the control structure model omits irrelevant detail from messaging protocols and interface conventions to focus on general control and feedback. This feedback-control modeling approach allows very complex systems to be handled in contrast to bottom-up techniques such as FMEA (Failure Modes and Effects Analysis) or HAZOP. In addition, it does not require a detailed design to examine failure modes as does Fault Tree Analysis. Another limitation with fault tree analysis is that it provides no assistance in determining what contributing events should be included in the tree. The completeness and relevance of the fault tree boxes is dependent entirely on the experience and insight of the safety engineer conducting the analysis. STPA provides analysts with guidance on what factors to consider and, with a minimum of training; the design engineers can perform STPA without expert safety engineering guidance. Attempting to use any of the traditional hazard analysis techniques on BMDS would have been futile.

In contrast, STPA does not require a detailed design nor does it examine all components in detail. Instead, the analysis is top-down, examining first the very high-level interactions among the primary components of the system. Many hazards can be completely resolved in this way. In other cases, the hazards arising from interactions among components and individual component behavior will require looking in more detail at the individual components in Figure 3. However, the behavior that must be examined at lower levels is identified and limited by the high-level analysis results.

In figure 3, the command authorities control system operators by issuing guidance, providing training, and establishing tactics, techniques, and procedures (TTPs). Command authorities receive feedback in the form of reports and performance during training exercises. Operators control the Fire Control software (system) by issuing commands and receive feedback from displays and aural alerts. The Fire Control Software controls other BMDS software and hardware by sending messages and commands and receives feedback in the form of measured values and status information.

A typical engagement would begin with a warning from the Early Warning System. The fire control system would then task the radar to track the target. Once the system has adequate data to plan an intercept, the operators direct the system to engage its target. The fire control software then develops interceptor tasking and sends it to the launch station. More generally, the operators control the behavior of the fire control software by directing the software to change operating modes (between test, exercise, and live operations). The launch station is responsible for the interface with the interceptor, transforming the interceptor tasking into a task load for the flight computer and controlling the launch sequence. The flight computer receives the task load, follows the launch sequence under control of the launch station, and ignites the rocket motor.

At the lowest level of the control structure, the flight computer is responsible for arming and safing the interceptor hardware. Feedback to the flight computer is provided in the form of Built-in Test (BIT) results and the status of whether the hardware is safe or armed. The interceptor simulator is used to perform BIT and test the readiness of the system at any point in time.

STPA analysis applies equally to hardware, software, or human commands. Some examples of unsafe control actions are:

1. The flight computer could omit the required control action of safing the interceptor hardware if a design error in the interface allowed the launch station to remove power from the flight computer before the abort sequence completes.
2. The launch station could issue an unsafe control action if it commands a real interceptor to arm when it is communicating with an interceptor simulator.
3. The operators might provide a correct control action too late if they transition to weapons hold too late, that is, after an interceptor has been inadvertently launched.

4. The command authority might contribute an inadvertent launch if a reduction in training to maintain operator proficiency increases the risk that the operators cannot act to prevent it.

The table below shows a snippet from a UCA table for the Fire Enable command, which is sent by the fire control software to the launch station. Only the conditions in red are related to the inadvertent launch command.

HAZARD: Inadvertent Launch

Control Action	Not providing causes hazard	Providing causes hazard	Too early/too late, wrong order	Stopped too soon/ applied too long
Fire Enable	Launch will not take place	Will progress to a launch sequence if interceptor tasking provided	LATE: Delays ability to process launch sequence EARLY: Can progress to launch sequence when … OUT OF SEQUENCE: Disable comes before the enable	N/A

If the Fire Enable command is provided to a launch station when there is no active threat (the red entry in column 3), the launch station will transition to a state where it accepts interceptor tasking and can progress through a launch sequence. In combination with other incorrect or mistimed control actions (related to interceptor tasking), this could contribute to an inadvertent launch. A Fire Enable command sent too early could open a window of opportunity for inadvertently progressing toward an inadvertent launch, similar to an incorrect fire enable. The degree of risk this UCA contributes depends both on the likelihood of the inadequate control and how early the control action is carried out. In the worst case, a Fire Enable command might be out of sequence with the Fire Disable command. If this is possible in the system as designed and built, the system could be left capable of processing interceptor tasking and launching when not intended.

Using STPA. unsafe interactions among control actions can be identified. For example, when a Fire Enable is sent, the fire control software might discover that the track contains an object that is not a threat and then issue a Fire Disable command: If the two commands arrive in the wrong order, inadvertent (unsafe) launch can occur.

Table 1 shows some examples of causal scenarios produced for the FMIS example for the unsafe control action “Fire Enable Provided Incorrectly” shown below:

Table 1: Some example scenarios

UCA: If the Fire Enable command is provided to a launch station incorrectly, the launch station will transition to a state where it accepts interceptor tasking and can progress through a launch sequence. In combination with other incorrect or mistimed control actions (see interceptor tasking), this could contribute to an inadvertent launch.
Scenario 1: The fire control computer is intended to send the Fire Enable command to the launch station upon receiving a Weapons Free command from an FMIS operator and while the fire control

system has at least one active track. According to the requirements and design specifications, the handling of the Weapons Free command is straightforward. But what makes a track active? Activity criteria are specified by the FMIS operators according to their operational procedures. The software supports declaring tracks inactive after a certain period with no radar input, after the total predicted impact time for the track, or after a confirmed intercept. It appears one case was not carefully considered: if an operator deselects all of these options, no tracks will be marked as inactive. Under these conditions, the inadvertent entry of a Weapons Free command would send the Fire Enable command to the launch station immediately, even if there were no threats to engage currently tracked by the system.

Scenario 2: The FMIS system undergoes periodic system operability testing using an interceptor simulator that mimics the interceptor flight computer. STPA identified the possibility that commands intended for test activities could be sent to the operational system. As a result, the system status information provided by the launch station includes whether the launch station is connected only to missile simulators or to any live interceptors. If the fire control computer detects a change in this state, it will warn the operator and offer to reset into a matching state. However, there is a small window of time before the launch station notifies the fire control component of the change during which the fire control software might send a Fire Enable command intended for test to the live launch station.

Neither of these causal scenarios involve component or subsystem failures. In both cases, all the components involved were operating exactly as intended; however, the complexity of their interactions led to unanticipated system behavior. Of course, component failure can also be a cause of unsafe control over a system's behavior, and STPA does identify such failures. But only the failures that are critical are considered, not all potential failures (as in FMEA).

In the official STPA analysis of BMDS, new and previously unknown scenarios were identified and recommendations generated to mitigate them. Such recommendations may involve hardware or software design changes, changes to maintenance and test procedures, and workarounds for the system operators. The costs (both in terms of dollars and time) to fix the hazardous scenarios (i.e., the scenarios that could lead to a hazardous system state) identified by STPA were considerable. As stated, if STPA had been performed early in the process, these costs would have been eliminated.

Summary

The invention of a new term, "systems of systems" is unnecessary as the term "system," as defined in system theory, already incorporates all the supposedly new properties. Worse, the new term is misleading and causing confusion. STPA can be used to perform a hazard analysis on all these supposedly different types of systems.

In this paper, the U.S. BMDS is used as an example. The hazard analyses performed on the individual systems combined to create BMDS (which therefore became subsystems of BMDS) cannot be somehow "combined" to perform a hazard analysis for BMDS. BMDS is a new system where the combined emergent behavior and hazards are different than the individual systems that have been composed to create this new system. Thinking of this as a system of systems provides no advantages and is not necessary to analyze BMDS. It simply confuses matters and makes the solution more obscure. A system hazard analysis and a system-of-system hazard analysis are the same thing, and thus both are not needed.