28th annual INCOSE international symposium
Washington, DC, USA
July 7 - 12, 2018

Safety Analysis in Early Concept Development and Requirements Generation[1]

Nancy G. Leveson
Massachusetts Institute of Technology
77 Massachusetts Ave, Cambridge MA 02139
617-258-0505
http://leveson@sunnyday.mit.edu
leveson@mit.edu

Abstract. This paper shows how a new hazard analysis technique, STPA (System Theoretic Process Analysis), can be used to generate high-level safety requirements early in the concept development phase that can then assist in the design of the system architecture. These general, system-level requirements can be refined using STPA as decisions are made. The process goes hand-in-hand with design and the rest of the lifecycle as STPA can be used to provide information to assist in decision-making throughout the development and even operations phases. STPA also fits into a model-based engineering process as it works on a model of the system (which is also refined as design decisions are made) although that model is different than the architectural models usually proposed for model-based system engineering today. The process promotes traceability throughout the development process so decisions and designs can be changed with minimum requirements for redoing previous analyses. Finally, while this paper describes the approach with respect to safety, it can be applied to any emergent system property.

Early Concept Exploration and Development

Early concept development, shown at the top left of the V-model (Figure 1), is the first step in the usual system engineering process. While the label may differ in variants of the model, this stage includes such activities as stakeholder and user analysis (needs analysis), customer requirements generation, regulatory requirements review, feasibility studies, concept and tradespace exploration, and establishment of criteria for evaluation of the evolving and final design. Toward the end of this step, there may be development 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 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 and maintain. 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.

[1] Submitted to INCOSE 2018

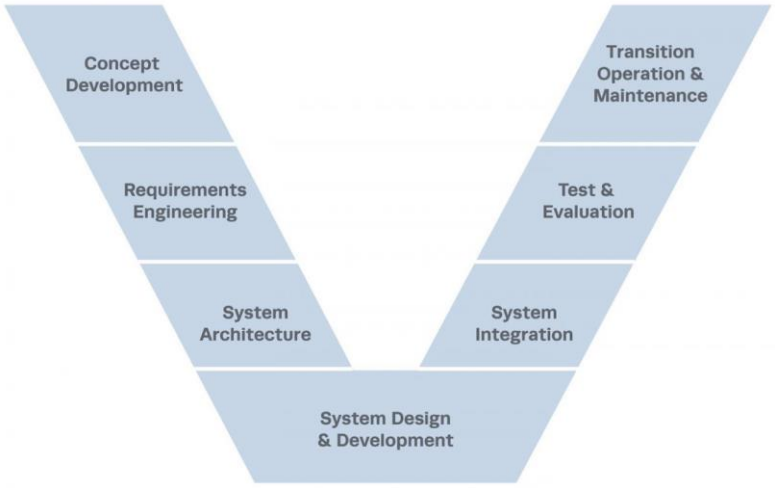

Figure 1. The V-Model for System Engineering

Safety and security concerns are often inadequately considered in early concept exploration. Figure 2 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.

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 components. Frola and Miller (1984) claim 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. To fix this problem, it is necessary to integrate safety and security into concept analysis and system requirements generation. This paper only covers safety due to paper length restrictions, but security can be handled in a similar fashion.

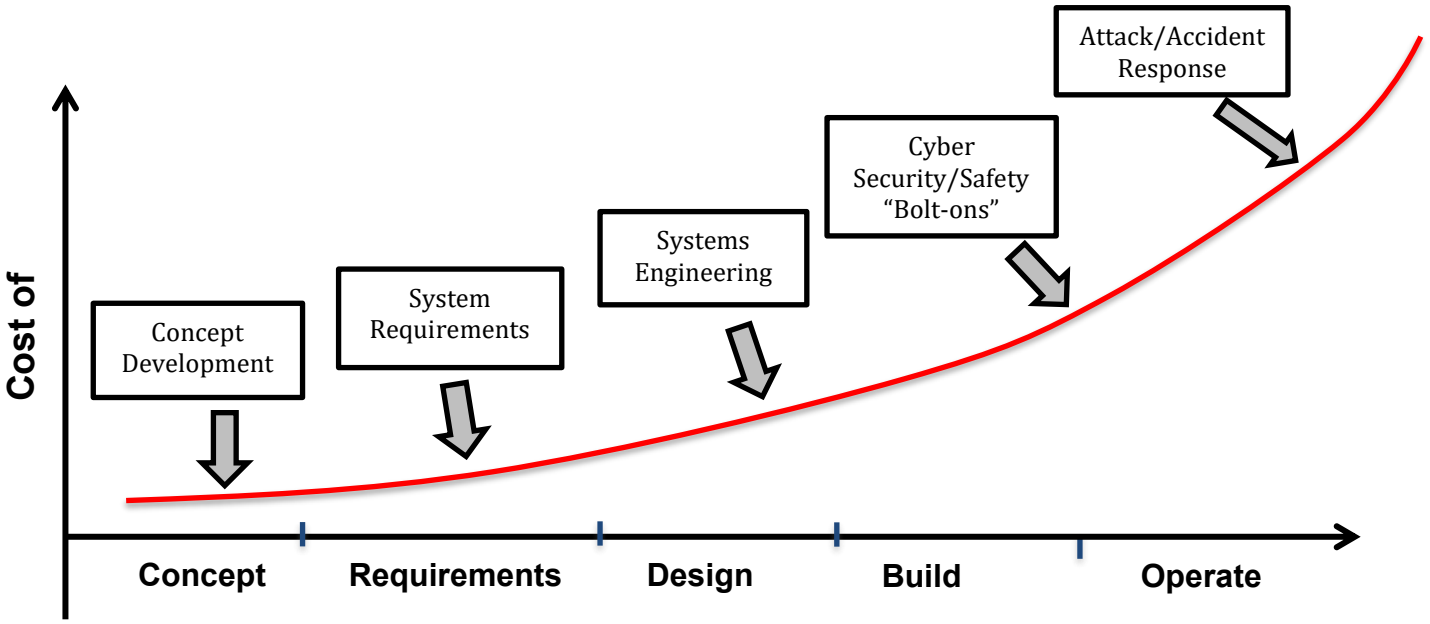

Figure 2. Safety and security need to be built into the system from the beginning of development [Young, 2017]

Traditional Preliminary Hazard Analysis (PHA)

While some consideration of safety is usually included in early concept analysis, usually in the form of a Preliminary Hazard Analysis (PHA), the goal is often to limit the effort necessary for later development and to predict general system risk and not to generate functional safety requirements and design recommendations.

The 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 the author's experience, architectures are often developed before the system requirements are identified. This inverted sequence raises the likelihood that the architecture will not be optimized and sometimes not even appropriate for the system goals.

The specific formats for PHA differ, but Table 1 shows a typical PHA table [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 when the system is software-intensive. Historical information is not useful when the system differs significantly from past systems. The result is often incorrectly dismissing specific hazards early in the system development process as "marginal" or extremely unlikely [Abrecht 2016, Abrecht et.al. 2016]

Table 1. A typical PHA format

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]

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 for use this early in development [Leveson 1995].

An alternative often used in the aviation community is to generate probabilistic requirements to be satisfied by the designed system [SAE 1996]. 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 1996]. 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 [Leveson 2012].

This paper describes a new approach to hazard analysis that can start in the early concept stage of development and generates the system and component functional safety requirements. It is illustrated on an aircraft ground braking system.

STAMP and STPA

STPA (System-Theoretic Process Analysis) is a relatively new hazard analysis technique based on an extended model of accident causation called STAMP (System-Theoretic Accident Model and Processes) [ESW].

Traditional hazard analysis methods assume a model of causality where accidents are conceived as being caused by a chain of component failure events, i.e., component A fails, which causes component B to fail, which leads to the failure of component C, which causes the actual loss event. For example, the brakes fail after the aircraft lands, the pilot is unable to decelerate in time to avoid an obstacle, and the aircraft crashes into the obstacle. This simple failure-based model is no longer adequate for the complex, software-intensive systems being built today where accidents may be caused by system design errors leading to interactions among components, none of which may have failed [Leveson 2012].

STAMP is based on system theory [see, for example, Checkland 1986], which was created to handle complex systems. STAMP extends the chain-of-failure causality model to include both component failure and unsafe interactions of system components. In STAMP, safety is an emergent[2] system property that arises when the components of a complex system interact with each other within a larger environment. A set of constraints (requirements) related to the behavior of the system components (physical, software, human, and social) enforces the safety property. For example, typical safety constraints on system operation are: aircraft and automobiles must never violate minimum separation standards, medical devices must not provide a harmful level of medicine, toxic chemicals/radiation must not be released from a plant, batteries must never experience thermal runaway, aircraft must have sufficient lift to remain airborne unless landing, nuclear materials must never get into the wrong hands.

Accidents occur when the system component interactions violate these constraints. The goal, then, is to control the behavior of the components and system as a whole to ensure that the safety constraints are enforced in the operating system. Rather than focusing only on preventing accidents by increasing component reliability and treating safety as a component failure problem, safety using STAMP is treated as a dynamic control problem that enforces the safety constraints where both component failures and component interactions must be controlled.

STPA (System-Theoretic Process Analysis) is a new hazard analysis technique based on the STAMP extended model of accident causation [Leveson 2012]. 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.
- STPA can be started in early concept analysis to assist in identifying safety requirements and constraints before any design or architecture exists. The STPA generated safety requirements 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 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.

[2] *Emergence* is a basic concept in system theory. Emergent system properties are not in the individual system components but *emerge* from the interactions among the components. The familiar phrase “the whole is greater than the sum of the parts” is derived from the concept of emergence.

- STPA includes software and human operators in the analysis, ensuring that the hazard analysis includes all potential causal factors in losses.
- The models developed for the STPA analysis provide documentation of system functionality (vs. physical or logical design) that is often missing or difficult to find in the documentation for large, complex systems.
- STPA generates both safety and security requirements and, in fact, requirements for any emergent system property. In this paper, we are concerned with not the usual information security but functional security (ensuring the functions of the system are not negatively affected by hostile actions).
- STPA can be easily integrated into any system engineering process and into model-based system engineering. In this case, however, the models are functional models rather than simply physical and logical models.

STPA is described and illustrated using an aircraft example in the rest of the paper. The example comes from a previous STPA aircraft hazard analysis [Leveson et.al. 2014]

Illustrating STPA with an Aircraft Example

Important activities in early concept analysis include identifying the stakeholder needs, system goals, and high-level system requirements. STPA starts with identifying stakeholder needs and goals with respect to safety. The next steps in STPA are to (1) build a high-level functional control structure that will be used for the rest of the STPA analysis (2) identify potential unsafe control actions, (2) generate high-level functional requirements from the unsafe control actions, and (3) generate causal scenarios for the unsafe control actions and use these to create more detailed system and component functional safety requirements. The scenarios include not only component failures but also additional factors such as direct and indirect interactions among system components (which may not have "failed"). The identified causal scenarios serve as the basis for developing system and component safety requirements and constraints. These requirements can be used in the design of the system architecture, in early design decisions, and in creating a concept of operations as well as in guiding the rest of the system development.

As later design decisions are made throughout the entire development process, the STPA analysis can be iterated and refined to support tradeoffs and decision making about the system and component design. Traceability is provided from any low-level safety related design decision back to the first steps in the concept analysis process and from the high-level concept analysis down to detailed design decisions.

Starting the Early Concept Analysis

The goal of the STPA analysis is to identify hazardous behaviors so they can be eliminated or controlled in the system design. These hazardous behaviors are used to identify the behavioral (functional but not probabilistic) safety requirements for the various system components, including the software and human operators.

The process starts with the stakeholders specifying 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 in this paper 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

Hazards are defined as they are in System Safety, that is, as system states or sets of conditions that, when combined with some set of environmental worst-case conditions, will lead to an accident or loss event[3] [Leveson, 1995; Leveson, 2012]. 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.

In STPA, humans, such as operators, maintainers, and even managers, are included as part of the system that is analyzed. STPA thus provides a structured method to identify human errors influenced by the system design, such as mode confusion and loss of situational awareness leading to hazards as well as the hazards that could arise due to loss of synchronization between the actual automation state and the operator's mental model of that state.

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

H1: Insufficient thrust to maintain controlled flight
H2: Loss of airframe integrity
H3: Controlled flight into terrain
H4: An aircraft on the ground comes too close to moving or stationary objects or inadvertently leaves the taxiway
H5: etc.

H4 is the most relevant for the example in this paper**.** 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:

Deceleration Hazards:

H4-1: Inadequate aircraft deceleration upon landing, rejected takeoff, or taxiing
H4-2: Deceleration after the V1[4] 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 (SCn) associated with these hazards are a simple restatement of the hazards in or 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 (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

[3] Sometimes a hazard is defined as a condition that could lead to 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 taking off could lead to an accident later on, but we cannot build a useful system where aircraft never take off.
[4] The V1 point is that point in the takeoff sequence where it is more dangerous to stop the takeoff than to continue

(H4-4)

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. 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.

Building a Model of the Functional Control Structure

After identifying the accidents, the system safety hazards to be considered, and the system-level safety requirements (constraints), the next step in STPA is to create a model of the aircraft functional control structure. The STPA analysis is performed on this functional control structure model, which basically uses standard feedback control loops.

Figure 3 shows a basic control loop. The controller issues control actions or commands to the controlled process: for example, in a car the control actions may include *brake* or *accelerate.* The control actions are generated by the controller using a control algorithm (for software) or a decision-making process for humans. In turn, the process of generating control actions uses information about what the controller thinks is the current state of the system, called here the *Process Model*. In a human, the process model is often called the "mental model." Feedback from the controlled process to the process is used to update this process model.

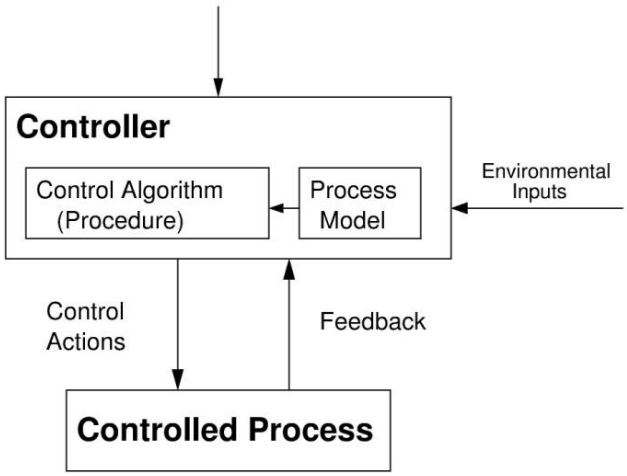

Figure 3. The basic feedback-control loop used in functional control structures.

The process model plays an important role in STPA. In systems theory, every controller contains a model of the controlled process. This process model or mental model includes assumptions about how the controlled process operates and the current state of the controlled process. It is used to determine what control actions are necessary to keep the system operating effectively and safely.

Accidents in complex systems often result from inconsistencies between the model of the process used by the controller and the actual process state, which results in the controller providing unsafe control actions. For example, the autopilot software thinks the aircraft is climbing when it really is descending and applies the wrong control law; a military pilot thinks a friendly aircraft is hostile and shoots a missile at it; the software thinks the spacecraft has landed and turns off the descent engines prematurely [JPL 2000]; the air traffic controller does not think two aircraft are on a collision course and therefore does not provide advisories to the aircraft to change course; or a

human or automated controller thinks that a valve is open when it is actually closed and does not issue a necessary *open valve* command [Kemeny 1979].

Part of the challenge in designing an effective safety control structure (and thus a safe system) is providing the feedback and inputs necessary to keep the controller's model consistent with the actual state of the controlled process. Identifying potential paths to accidents and losses involves determining how and why the controls could be ineffective in enforcing the safety constraints on system behavior; often this is because the process model used by the controller is incorrect or inadequate in some way, which in turn results from incorrect (including intentionally compromised for security-related incidents) or delayed feedback . The causes of such an inconsistency are identified by STPA.

A large percentage of accidents involving software can be explained by inaccurate process models. Analyzing what is needed in the software process model is an important part of the safe design for a system containing software. The same is true for accidents related to human errors. STAMP provides a way of identifying safety-critical information and potential operator errors and their causes, so they can be eliminated or mitigated.

There are four types of unsafe control:

- Not providing the control action under specific conditions causes a hazard
- Providing the control action under specific conditions causes a hazard
- Providing the control action too soon, too late, or out of sequence (relative to another control action) causes a hazard
- Stopping the control action too soon or applying it too long causes a hazard (applicable only to continuous control actions).

These four types of unsafe control play an important role in STPA in identifying unsafe control actions, as shown in the next section.

Figure 4 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), 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.

While a general control structure that includes the entire socio-technical system, including both development and operations, can be used in the analysis, in this example we consider only the aircraft itself. 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 or automation. No distinction is necessary at this point of development.

The role of the pilot, as shown in the Figure 4 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 environment in which they are operating. Many pilot errors can be traced to flaws in their understanding of how the automation works or of the current state of the automation.

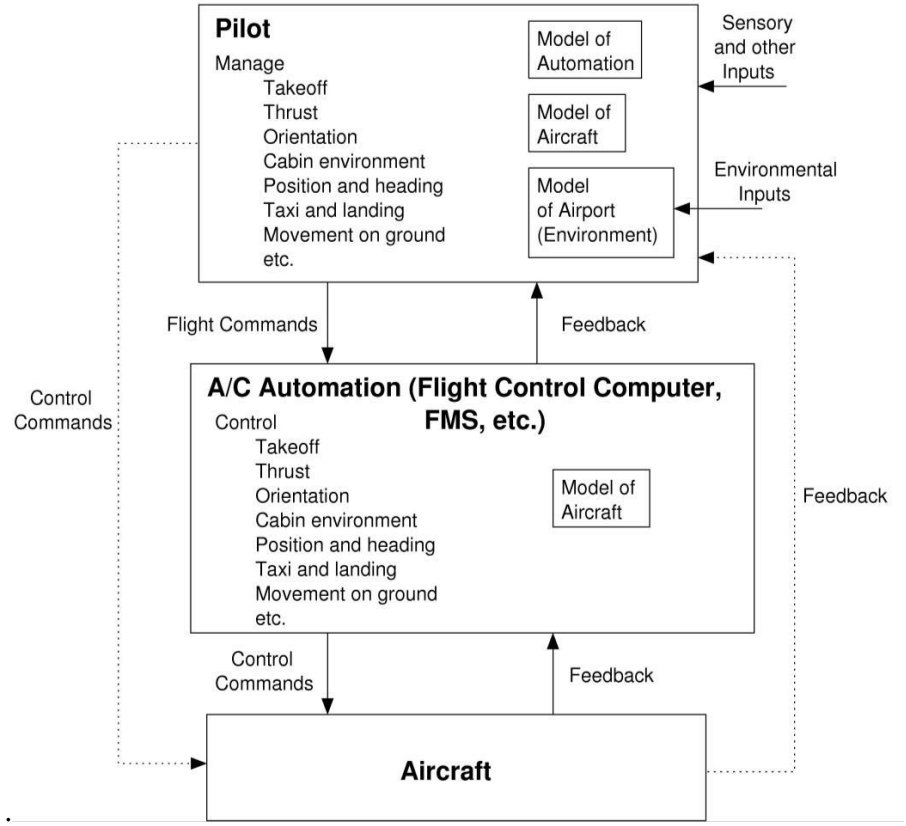

Figure 4. A High-Level Control Structure at the Aircraft Level

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 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).

Figure 5 zooms in on the control structure model for the ground control function, which is the focus of the example in this paper. There are three basic physical components being controlled: the reverse thrusters, the spoilers, and the wheel braking system. 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.

The intent of the model in Figure 6 is to show the “functional” structure of the wheel brake system, like the other control structures presented so far, without any assumptions about the physical implementation. STPA starts without a specific design solution to potential problems. Instead, it starts from the basic required functional behavior and identifies the ways that that behavior can be hazardous. Designers can later decide on particular design solutions, such as redundancy, necessary to satisfy the safety requirements derived through this analysis.

Figure 6 includes an autobraking function. Pilots can preset an autobrake before they land if they are going to be busy with other activities after landing. If set, the autobrake will engage the wheel brakes at the specified time.

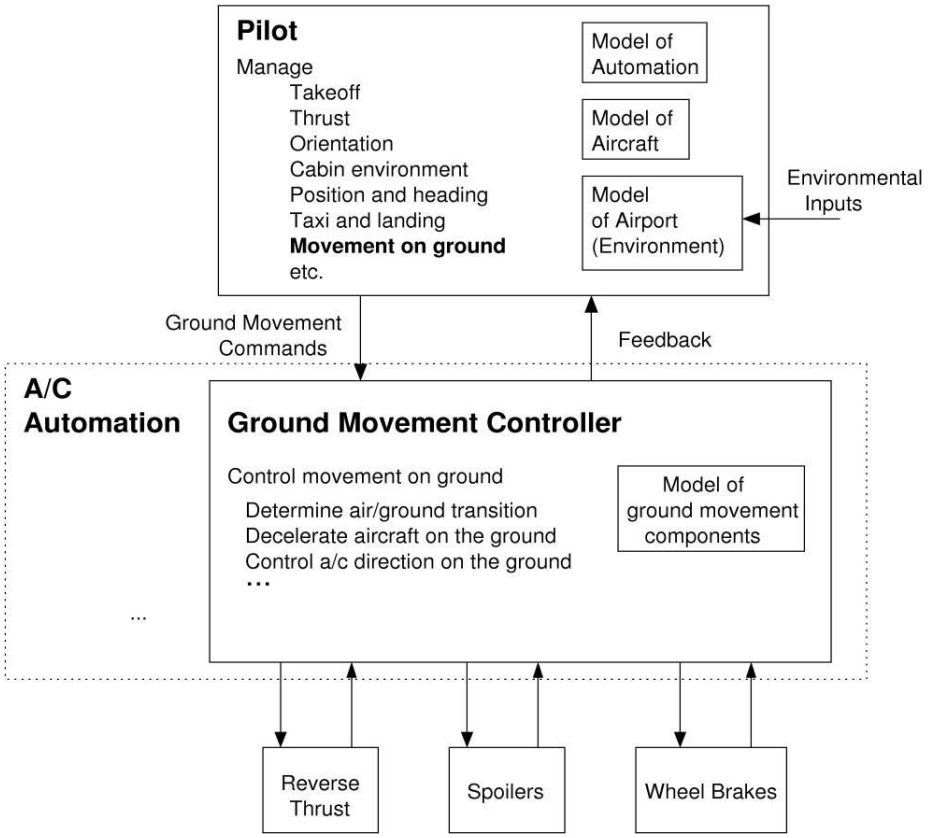

Figure 5. Deceleration Control Structure

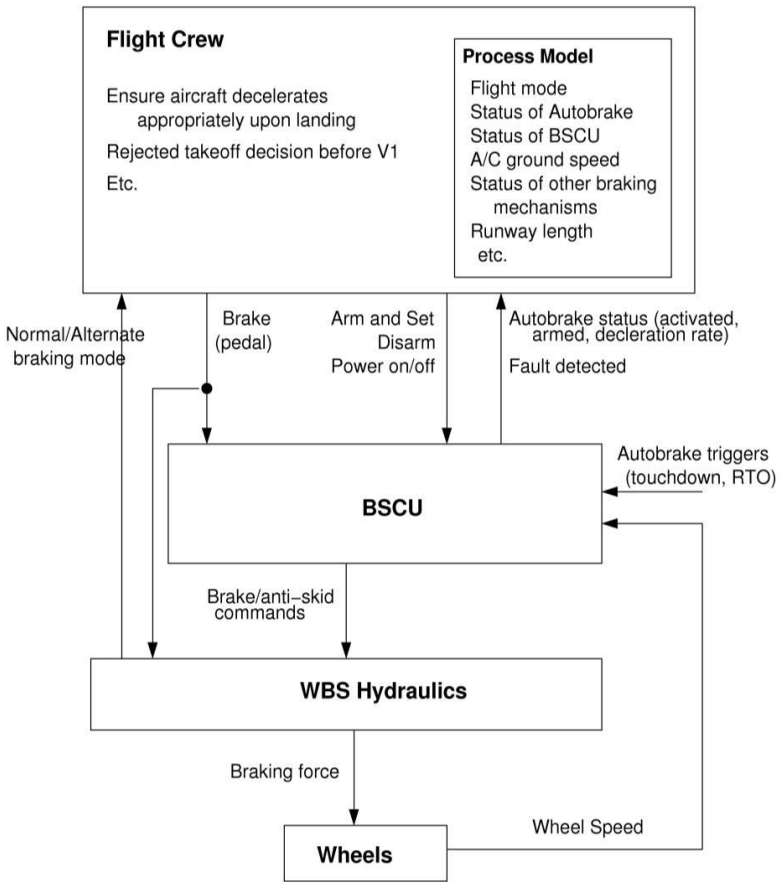

Figure 6. The Control Structure for the Wheel Braking System

Identifying Unsafe Control Actions

In STAMP, safety is treated as a control problem. In order to generate the scenarios leading to losses, the first step is to identify unsafe control actions. At this stage in the analysis, it is immaterial whether control actions are provided manually or automatically. The purpose is to define the hazardous control actions from any source. Automated tools based on a mathematical formalization of STPA have been created to assist in this part of the analysis [Thomas, 2013], but they are beyond the scope of this paper.

A table is a convenient way to document the unsafe control actions. Table 2 shows a few examples of unsafe control actions that can be provided by the crew, Table 3 shows some examples for actions that can be given by the BSCU Autobrake controller. The entries in the tables include both the control action (found in the control structures) and the conditions under which that control action will be hazardous. The first column lists control actions that can be given by the controller and the four following columns list how those control actions could be hazardous for the four general types of unsafe control described in the previous section. References are provided to the associated hazard in order to maintain traceability. Where design information is needed to provide more detailed requirements, "TBD" is used to note that the information will need to be determined later in the design process.

The table only shows unsafe control actions. Hazards that result when a safe control action is provided but not followed or executed—which is the major focus of most hazard analysis techniques—are identified later.

Table 2. Unsafe Control Actions for the Flight Crew

Control Action By Flight Crew:	**Not providing causes hazard**	**Providing causes hazard**	**Too soon, too late, out of sequence**	**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 insufficient braking), leading to overshoot [H4-1, H4-5]	CREW.1b1 Manual braking provided with insufficient pedal pressure, resulting inadequate deceleration during landing [H4-1, H4-5	CREW.1c1 Manual braking applied before touchdown causes wheel lockup, loss of control, tire burst [H4-1, H4-5]	CREW.1d1 Manual braking command is stopped before safe taxi speed (TBD) is reached, resulting in overspeed or overshoot [H4-1, H4-5]

Table 3: Unsafe Control Actions for the Brake System Control Unit Autobrake Controller

Control Action by BSCU	**Not providing causes hazard**	**Providing causes hazard**	**Too soon, too late, out of sequence**	**Stopped too soon, applied too long**
BSCU.1 Brake command	BSCU.1a1 Brake command not provided during RTO (to V1), resulting in inability to stop within available runway length [H4-1, H4-5]	BSCU.1b1 Braking commanded excessively during landing roll, resulting in rapid deceleration, loss of control, occupant injury [H4-1, H4-5]	BSCU.1c1 Braking commanded before touchdown, resulting in tire burst, loss of control, injury, other damage [H4-1, H4-5]	BSCU.1d1 Brake command stops during landing roll before TBD taxi speed attained, causing reduced deceleration [H4-1, H4-5]

The unsafe control actions not only are used to create the causal scenarios, but they can also be used to create requirements and safety constraints on the system and component design and implementation. For example, a safety constraint on the pilot might be that manual braking commands must be provided to override Autobrake in the event of insufficient Autobraking. Such constraints on humans clearly are not enforceable in the same way as constraints on physical components (i.e., by physical design features), but they can be reflected in the design of required pilot operational procedures, in training, and in performance audits. Some requirements that are considered to be error-prone or unachievable by human factors experts might result in changes in the braking system design.

A few example requirements for the flight crew and the BSCU derived from the unsafe control actions are shown in Table 4. Note there is traceability to the specific unsafe control actions from which the requirements are generated as well as documentation of the rationale for each requirement.

Table 4. 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

Generating Causal Scenarios

Once the unsafe control actions have been identified, their causal scenarios are generated. The causes of a hazard when safe control was provided but that control action was improperly executed or not executed by the controlled process are also generated. The process for generating these scenarios is beyond the scope of this paper, but some examples are provided in this section. All the scenarios can involve unsafe control actions and process model flaws across multiple controllers. Complete scenarios need not be limited to any single component. In fact, an unsafe control action by one controller might indirectly cause an unsafe control action by another controller.

UNSAFE CONTROL ACTION – CREW.1a1: Crew does not provide manual braking when there is no Autobraking and braking is necessary to prevent H4-1 and H4-5.

Scenario 1: Crew incorrectly believes that the Autobrake is armed and expect the Autobrake to engage (process model flaw). Reasons that their process model could be flawed include:

a) The crew previously armed Autobrake and does not know it subsequently became unavailable, AND/OR

b) The feedback received is adequate when the BSCU Hydraulic Controller detects a fault. The crew would be notified of a generic BSCU fault but they are not notified that Autobrake is still armed (even though Autobraking is no longer available), AND/OR

c) The crew is notified that the Autobrake controller is still armed and ready, because the Autobrake controller does not detect when the BSCU has detected a fault. When the BSCU detects a fault it closes the green shut-off valve (making Autobrake commands ineffective), but the Autobrake system itself does not notify the crew.

d) The crew cannot process feedback due to multiple messages, conflicting messages, alarm fatigue, etc.

Possible new requirements for S1: The BSCU hydraulic controller must provide feedback to the Autobrake when it is faulted and the Autobrake must disengage (and provide feedback to crew). Other requirements may be generated from a human factors analysis of the ability of the crew to process the feedback under various worst-case conditions.

UNSAFE CONTROL ACTION – BSCU.1a2: Brake command not provided during landing roll, resulting in insufficient deceleration and potential overshoot

Scenario 1: Autobrake believes the desired deceleration rate has already been achieved or exceeded (incorrect process model). The reasons Autobrake may have this process model flaw include:

a) If wheel speed feedback influences the deceleration rate determined by the Autobrake controller, inadequate wheel speed feedback may cause this scenario. Rapid pulses in the feedback (e.g. wet runway, brakes pulsed by anti-skid) could make the actual aircraft speed difficult to detect and an incorrect aircraft speed might be assumed.

b) Inadequate external speed/deceleration feedback could explain the incorrect Autobrake process model (e.g. inertial reference drift, calibration issues, sensor failure, etc.).

Possible Requirement for S1: Provide additional feedback to Autobrake to detect aircraft deceleration rate in the event of wheel slipping (e.g. fusion of multiple sensors)

STPA considers not only inadvertent unsafe control actions as causes of hazards but can also handle those arising from intentional security-related actions such as intentionally changing the process model to make the controller provide an unsafe control action. For example, an intruder changes the feedback to the crew in Scenario 1 above to indicate to the crew that the autobrake is armed when it is not. Something similar happened in the Stuxnet worm attack on the Iraqi nuclear program.

Refining the System and Component Design Requirements

As design decisions are made, the STPA analysis is 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 7. 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 related to the BSCU hydraulic controller when controlling the three individual valves included in the refined design:

HC-R1: The HC must not open the green hydraulics shutoff valve when there is a fault requiring alternate braking [HC.1b1]

Rationale: 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]

Rationale: 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]

Rationale: Crew would be unaware that uncommanded braking was being applied.

Once again, traceability is maintained.

Scenarios for how these new unsafe commands 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 [Horney 2017]. The process continues throughout system development.

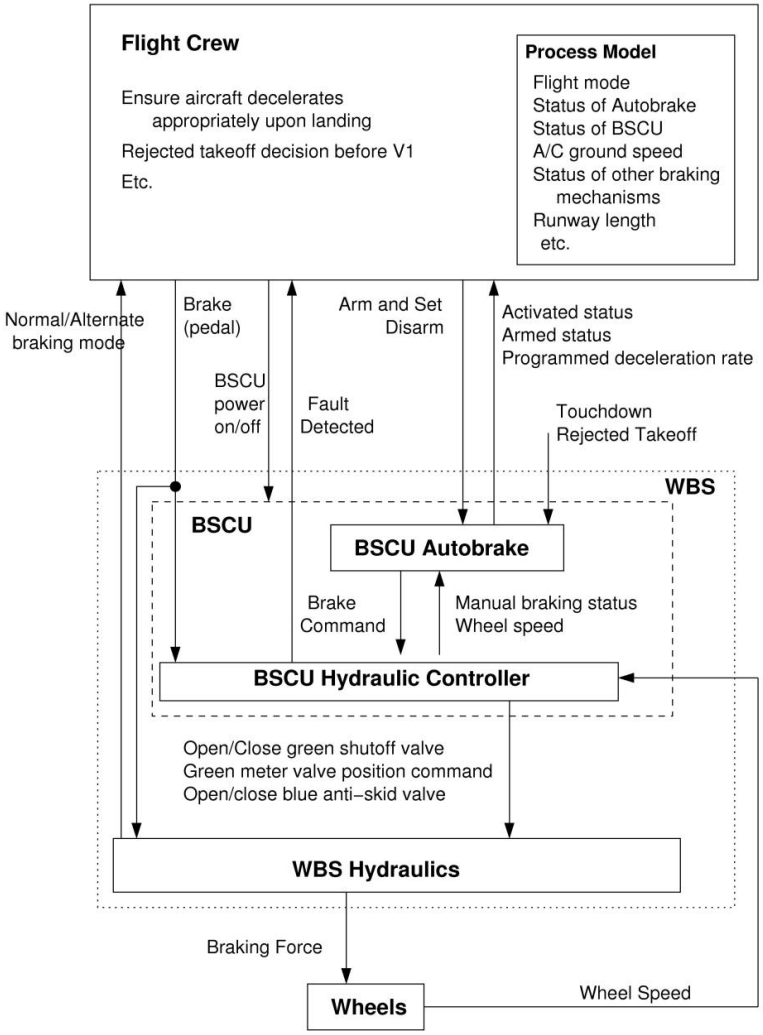

Figure 7. A More Detailed Version of the Wheel Brake System Control Structure

Summary and Conclusions

While STPA has been extensively used and evaluated in later stages of development, the use in early concept analysis has only recently been tried for a complex new military system in the very early concept development stage [Horney 2017]. This paper has described how this goal might be accomplished using STPA. Current use of safety analysis in concept development is primarily limited to assessing risk of system hazards in order to determine the emphasis to be placed on them during development. While this assessment might help management and government decision makers, it does not provide useful information for system designers and developers and is very likely incorrect. In aviation, probabilistic requirements are generated, but these are not very useful for software, which is a major component of aircraft today. STPA can generate functional safety requirements for systems as a whole and for the hardware, software, and human components of the

system. These requirements can be used to create an appropriate system architecture for satisfying the system requirements, including the safety requirements.

As development proceeds and the design is refined, the STPA analysis can be iterated with corresponding refinement of the requirements and controls designed to deal with hazards. Traceability is maintained throughout.

While safety is the focus of this paper, the same process can be used for any emergent system property.

References

Abrecht, Blake, 2016, 'Systems Theoretic Process Analysis Applied to an Offshore Supply Vessel Dynamic Positioning System', M.S. thesis, MIT, Cambridge, MA (US).

Abrecht, Blake, Arterburn, Dave, Horney, David, Schneider, Jon, Brandon, Abel, Nancy Leveson, 2016, A New Approach to Hazard Analysis for Rotorcraft, *AHS Specialists Meeting*, Huntsville, AL (US).

Checkland, Peter (1986), *Systems Thinking, Systems Practice*, John Wiley & Sons, New York, NY (US)

Frola, F.R. and Miller, C.O., 1984, 'System Safety in Aircraft Acquisition', Logistics Management Institute, Washington D.C. (US)

Horney, David, 2017, 'System-Theoretic Process Analysis and Safety-Guided Design of Military Systems, M.S. thesis, MIT, Cambridge, MA (US)

JPL Special Review Board, 2000, Report of the Mars Polar Lander and Deep Space 2 Missions, Jet Propulsion Laboratory, *Pasadena*, CA (US).

Kemeny, 1979, 'Report of the President's Commission on the Accident at Three Mile Island,' U.S. Government Printing Office.

Leveson, Nancy, 1995, *Safeware: System Safety and Computers*, Addison Wesley Publishers, New York, NY (US).

Leveson, Nancy, 2012, Engineering a Safer World, MIT Press, Cambridge, MA (US).

Leveson, Nancy, Wilkinson, Chris, Fleming, Cody, Thomas, John, 2014, 'A Comparison of STPA and the ARP 4761 Safety Assessment Process', MIT Technical Report, MIT, Cambridge, MA (US).

SAE, ARP 4761, 1996, '*Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment'*, RTCA (US)

Thomas, John, 2013, 'Extending and Automating a Systems-Theoretic Hazard Analysis for Requirements Generation and Analysis, Ph.D. thesis, MIT, Cambridge, MA (US)

Vincoli, Jeffrey, 2005, *Basic Guide to System Safety*, Wiley Inter-Science, New York, NY (US)

Young, William, 2016, Systems Theoretic Security Engineering, Ph.D. thesis, MIT, Cambridge, MA (US)

Biography

Nancy Leveson is Professor of Aeronautics and Astronautics at MIT. She is an elected member of the National Academy of Engineering (NAE). Prof. Leveson works in system and software engineering, system safety, and human-computer interaction. She has received many awards for her research, including the ACM Allen Newell Award for outstanding computer science research and the ACM SIGSOFT Outstanding Research Award. She has published over 200 research papers and is author of two books, "Safeware: System Safety and Computers" published in 1995 by Addison-Wesley and "Engineering a Safer World" published in 2012 by MIT Press. She consults extensively in many industries on the ways to prevent losses. Prof. Leveson served on the INCOSE Board of Directors from 1992-1996.