INCORPORATING NEW METHODS OF CLASSIFYING DOMAIN INFORMATION FOR USE IN SAFETY HAZARD ANALYSIS

Professor Nancy G. Leveson
Massachusetts Institute of Technology
Cambridge, MA

Major Daniel R. Montes
United States Air Force, Massachusetts Institute of Technology
Cambridge, MA

Professor Leia A. Stirling
Massachusetts Institute of Technology
Cambridge, MA

The increase of interacting humans and autonomous components in complex systems necessitates rigorous methods to classify domain information pertaining to controllers in the system. Systems-Theoretic Process Analysis (STPA) was developed at MIT as a method for identifying hazardous scenarios from a system design in order to generate functional system requirements to eliminate or control those scenarios. An STPA analysis, while systems-based and including human operators (e.g., pilots and air-traffic controllers) in the scenarios, is currently limited in the types of human contribution to accidents that it can identify (which are primarily related to situation awareness). This paper extends STPA in three ways: first, the analysis of the controller mental model was updated to include more system features; second, fundamental human-engineering considerations were added; and third, types and sources of decision-making influences that transfer from the planning cycle to the operations cycle were identified.

Humans play an important role in accidents (both positive and negative) and must be included in any hazard analysis performed during the development or field use of the system. Currently, safety investigators discuss human contributions in a simplistic way, if they include them at all (e.g., "pilot failed" or "pilot lost situation awareness"), and then they assign a probability to this "failure". Such analyses are not very useful in designing to prevent accidents.

The MIT Systems-Theoretic Accident Model and Processes (STAMP) is a use-centered systems-modeling approach to understanding and preventing accidents (Leveson, 2012). Systems-Theoretic Process Analysis (STPA) is a hazard-analysis technique based on STAMP that is used to investigate system designs in order to generate functional safety requirements. STPA goes beyond the tendency to simply state that a human failed and investigates errors in the human's mental model and errors in decision-making. This method, although advanced in terms of safety analysis, still oversimplifies the human's role in complex systems because it is currently posed similar to investigating a machine controller's model and decision algorithm. Human mental models contain more information about the system than a machine's and develop using more sources of feedback.

This paper extends STPA methods of generating hazardous scenarios by refining how the human (or intelligent controller) is considered in the analysis. The methodology now identifies more system information the controller might use to make decisions during the operation, considers human-specific controller characteristics (e.g., workspace and human variability), and identifies organizational influences to controller behavior that originate before the operation. An overview of STAMP will be presented, followed by STPA extensions for intelligent controllers. The new techniques presented are meant to analyze currently existing systems, although some aspects may apply similarly in concept development and design.

System Safety Modeling

STAMP was inspired by cybernetics (Wiener, 1965) and systems theory (Von Bertalanffy, 1968), as well as the U.S. system-safety standards that evolved during the development of long-range guided missiles (Department of Defense, 2012). Dekker (2006) describes two types of accident models in existence today. The first considers physical-component reliabilities (including people and software) and finds failures that chain together in time and/or space and lead to accidents. This type of model gives rise to fault-tree analyses and failure mode and effects analyses, for example. The second type of accident model, of which STAMP is an example, treats the prevention of undesirable losses as a top-level set of system requirements, and then it generates constraints to meet these goals through the

functional system behavior. STAMP treats safety as a control problem. The main concepts of STAMP are safety constraints, the hierarchical control structure, and process models (Leveson, 2012).

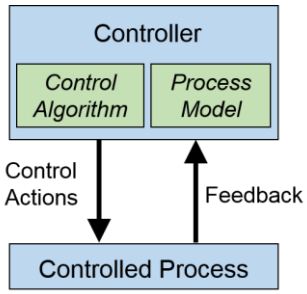

Figure 1. A basic control loop between functional entities in the safety control structure. Although a horizontal decomposition is not shown, disturbances, actions from other controllers, and communications would be modeled when appropriate.

Organizational stakeholders identify accidents[1], which are typically domain independent (e.g., environmental damage); then hazards[2] are identified, which are domain specific (e.g., chemical toxins exposed to the environment). The list of hazards is typically short because hazards may not include any engineering assumptions from the design (e.g., valve leaks and releases toxin). The prevention of each hazard becomes a *safety constraint*. Each hazard is mapped to one or more accidents. This mapping allows traceability from the findings of an ensuing hazard analysis (like STPA) all the way up to accidents.

The STAMP *hierarchical control structure* is an abstract model of the system design. It is a decomposition, starting at the top with legal/regulatory and organizational entities all the way to the lower-level components of the system operations. Higher levels have more responsibility, authority, and accountability than lower levels, and control-feedback relationships exist between levels. A general form (not pictured here) can be found in Leveson (2012). The control structure is a model of the functional system and not necessarily the physical structure. For example, suppose air-traffic control speaks to a drone operator through a UHF radio signal that travels from a control tower to the drone, is multiplexed and sent to the operator's ground control station via datalink band, and then demuxed into an audio channel in the operator's headset. While a physical schematic would detail the intricate connections just described, a STAMP control structure would show the tower personnel controlling the drone operator, who in turn would be controlling the drone.

In an STPA analysis, each level of the control structure is explored from the top down, and control relationships between entities are examined. A general control-feedback loop is shown in Figure 1. In "Step 1" of STPA, functional behaviors of a controller that violate safety constraints are identified as unsafe control actions (UCA), along with the system or environmental context in which they are hazardous. In "Step 2," causal scenarios that could produce each UCA are generated. This detailed analysis requires domain subject-matter experts (SME) because aspects of the physical design, hardware, software, and humans contribute to scenarios.

Currently, all the portions of the control loop—as well as any additional external information being used by the controller—are investigated to generate causal scenarios, including the controller itself and its *process model*. The process model is the controller's understanding of the states in the system it is trying to control. If the model states do not match the true system states, the controller could execute a UCA. In humans, this is called the mental model, although "process model" may be used generally. The following section discusses extensions to STPA that include refinements to the process model.

Extended Human Controller Analysis

Stringfellow (2010) and Thornberry (2014) previously elaborated on the human as a controller, with the former emphasizing that humans have a model of the organization, not just the controlled process, and the latter introducing a sequence for the Step-2 human-controller analysis. This extension will: 1) build on the existing sequence and add new sections, 2) refine the inquiries in some sections, and 3) introduce a method for identifying organizational influences on the controller. Figure 2 is the updated analysis sequence. It is *not* meant to be interpreted as an

[1] Accident: An undesired or unplanned event that results in a loss, including human, property, environmental, mission, etc.

[2] Hazard: A system state or set of conditions that, together with a particular set of environmental conditions, will lead to an accident.

information-processing (in-the-head) model of human cognition, but rather as a set of considerations that map the controller to the work domain. The structure of parts (a) through (e) is maintained from Thornberry, while (f) through (h) are new. Parts (a) through (c) have been refined.

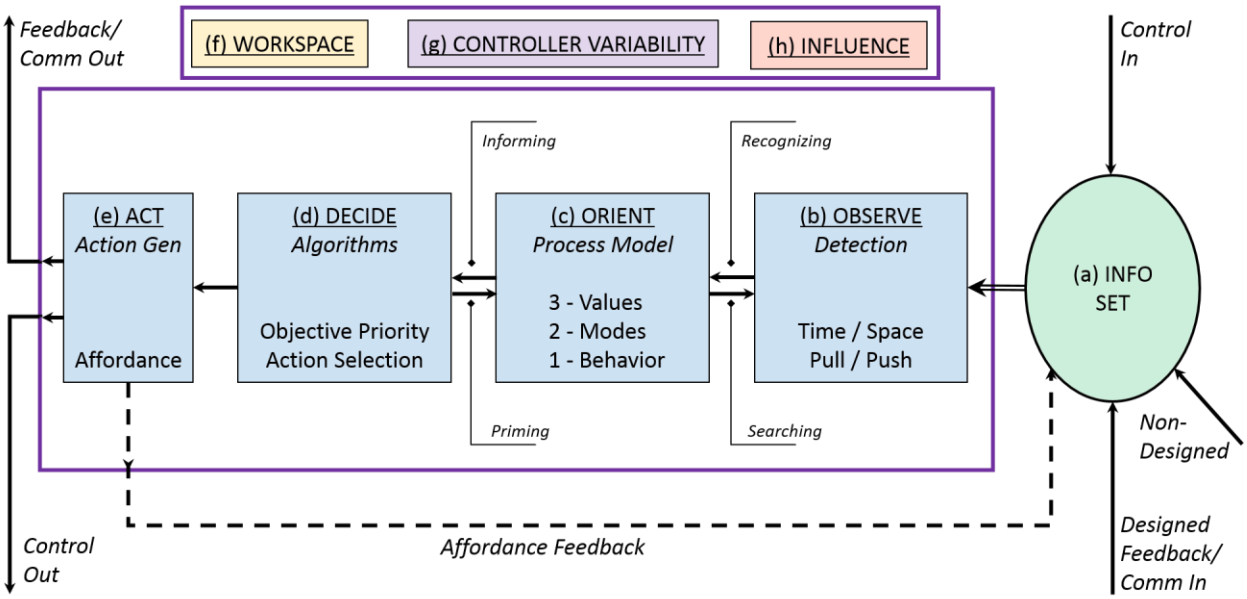

Figure 2. Extended intelligent-controller analysis sequence. Letters (a) through (h) denote areas for investigation.

sequence and add new sections, 2) refine the inquiries in some sections, and 3) introduce a method for identifying organizational influences on the controller. Figure 2 is the updated analysis sequence. It is *not* meant to be interpreted as an information-processing (in-the-head) model of human cognition, but rather as a set of considerations that map the controller to the work domain. The structure of parts (a) through (e) is maintained from Thornberry, while (f) through (h) are new. Parts (a) through (c) have been refined.

Part (a), the information set, corresponds to Dekker's "data availability" (2006) and helps identify all the information presented to the controller, including controls, feedbacks, and communications. This part has now been refined to explicitly differentiate between information available *as originally designed* to arrive at the controller and information outside original design intent that is being used by the controller regardless. An example of the latter would be a copilot looking at how the pilot's hands are displacing a traditional yoke instead of looking at the copilot's own flight displays, or two employees using an informal socio-organizational communication channel. The purpose of making the effort to delineate between design and non-design communications is crucial when new technologies and system upgrades threaten to change the nature of human-system interactions without properly documenting all the connections. Knowing that pilots were using "free" feedback (such as yoke movement) that goes away with an upgrade (such as fly-by-wire) is important.

Additionally, Thornberry (2012) emphasized that feedbacks the controller receives when an affordance is acted upon (such as a switch physically moving or a "bug" being set on an airspeed display) should be identified. This is important for similar reasons as non-designed information. For example, turning and removing a key from a classic (non-electronic) car ignition is sufficient feedback to the driver that the vehicle changed to a shutdown state; feedback from the controlled process (the car) was not required for the human to conclude the change had occurred.

Parts (b)-(e) in Figure 2 are named Observe, Orient, Decide, and Act after Boyd's O-O-D-A loop (2010). Part (b) corresponds to Dekker's "data observability" (2006) and performs inquiries on *if* and *how* data are attended to in time-space. A refinement here adds that data can be pushed to or pulled by the controller in several ways ranging as follows: the controller requests a controlled component or process for missing data, fetches already available data that is not yet displayed, refreshes an obsolete display, attends to a current display, or receives data immediately via her currently attended time-space (or through exogenous cueing). Boyd (2010) emphasized that *orient*—part (c)—is the most vital investigation area for analyzing decisions in a complex adaptive system. This section of the analysis maps to the *process model* in STAMP. The investigation here has been refined to include

three levels: behavior, modes, and values. *Behavior* represents how the controlled process is interacting with the mission environment. A behavior state variable might be a direct reading from the information set, or it might first be translated into a more useful variable (e.g., altitude and airspeed displayed as energy). A supervisor might be monitoring a lower-level controller which is managing the process behavior. In this situation the next level of the process model analysis (mode) becomes important.

Table 1.
Three types of modes (Leveson et al., 1997) and recommended inquiries that were added to the process model analysis.

Supervisory Structure	**The control relationships and communication links in the system hierarchy.**
	Which controllers currently have or share priority over each controlled component?
	Which controlled components may apply authority limits and under what circumstances? Can those limits be overridden? How will conflicts be decided (i.e., who should have the final authority?)
Component Operating Mode	**The set of algorithms that components under my control can use to exert control over their process(es).**
	What are the physical or logical assumptions and constraints associated with the component's current operating mode?
	What data in the information set is the controlled component using to inform its model?
	What input/and output format am I using with my controlled component(s)?
Mission Phase	**The specified set of related behaviors of the controlled system representing its operational state.**
	What mission phase is the system in (e.g., takeoff, cruise, etc.)
	Do all controllers know the current mission phase?
	Does a change in mission phase mode cause a change in supervisory structure and/or component operating modes (including input/output formats)?

A *mode* is a mutually exclusive set of system behaviors (Leveson, Pinnel, Sandys, Koga, & Reese, 1997). There are three types of modes: *supervisory structure*, *component operating mode*, and *mission phase*. Stringfellow's (2010) model of the organization, for example, would fall under supervisory structure. Table 1 presents definitions and a minimum set of recommended inquiries that have been added to investigate modes in the process model. *Authority limits*—worth mentioning—are a type of lockout or interlock that controlled components may exercise, by design, to ignore a received control request if they know it to be hazardous to the system. For example, a flight-control computer can limit the angle of attack the pilot demands, or a pilot can disregard an air-traffic control request if she sees visual traffic in the way. These limits must be carefully analyzed to make sure they do not prohibit behavior that might be necessary in some situations. In addition, there must be some determination of who should have the final authority in case of a conflict.

Values contains two lines of inquiry. The first is *external values*, which is an understanding of any values the controller personally maintains outside the system. An example would be the personal pressure behind the classic "get-there-it is" that might prompt a pilot to ignore system-derived objectives and rush a landing. The second is *value mapping*, which is the controller's understanding of how values at higher abstractions of the system's means-ends hierarchy (Rasmussen, Pejtersen, & Goodstein, 1994) map to objectives at the controller's level.

While STPA investigates control algorithms as part of Step 2, there is no specific methodology prescribed for human decision-making (d), and one is not recommended here. Figure 2 highlights that the mental model is affiliated with observation (searching-recognizing) as well as decisions (priming-informing), both which a human controller cognitively attends to. Part (e), appropriateness of response mappings, is not refined here.

Workspace inquiries (f) are new to the analysis and include climate, visual and auditory noise, work physiology (e.g., "pulling Gs"), anthropometric and ergonomic compatibility, and workload. *Controller variability* inquiries (g) are also new and include age, perceptual acuity, attention capability, natural disposition, health, injury/disabilities/disease, psychological/emotional conditions, fatigue/stress/sleep cycles, and drugs/medications. Findings in both (f) and (g) are human-specific considerations. In addition to being used to evaluate the operator

population for current systems and applications, they could be used to create criteria during design development to select a future population. Part (h), called *influence*, is new and it is discussed next.

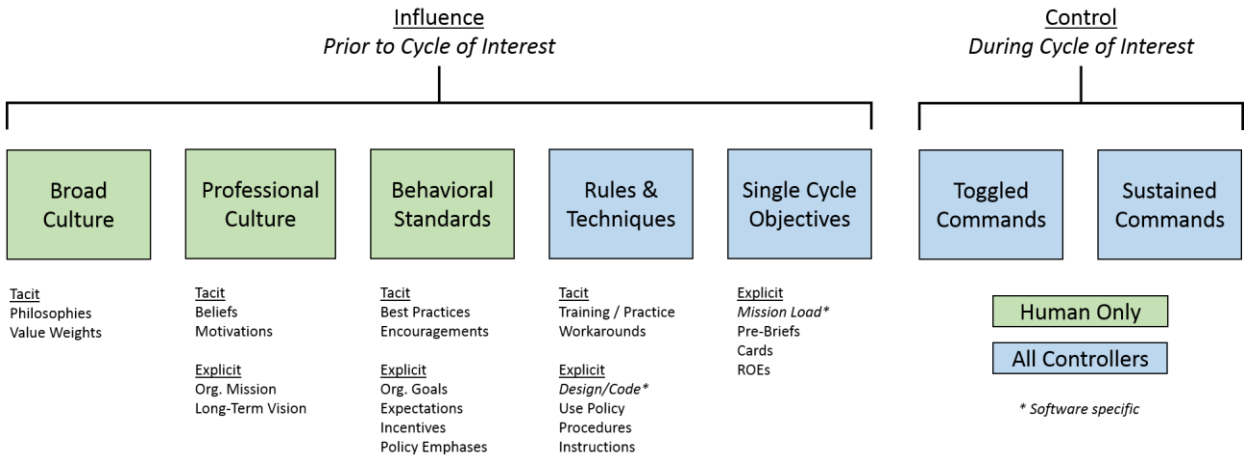

Figure 3. Sources of decision-making influences that evolve prior to the operation cycle. Green sections affect only humans.

Decision-Making Influences to the Operating Process

Influences can affect both human and machine (or software) controllers, although some are specific to only humans. The need to identify influences enforces the necessity of SME involvement during Step 2 of STPA. Socio-technical organizations contain large functional hierarchies, and often the lower levels of operation exhibit a smaller time constant (faster cycling) than higher-level managerial processes. Additionally, there may be planning or maintenance cycles that precede operating cycles, depending on the industry. Influences to operating controllers from these sources evolve before the operating cycle, and they are presented in Figure 3. Influences are controls, but because they do not occur in the real time of the operating cycle they are considered static during the operation (and thus can be incorporated into a section of the controller analysis).

Going from left to right in the figure, each consecutive block represents influences that evolve closer in time to the operating cycle. Influences can be unintentional. Conflicting policies or outdated procedures are an example of the wrong information making its way into the operating cycle. Sources of the influences can be explicit (formal, articulated, and codified) or tacit (not easily transferred via media). In a tacit example, operators while on their lunch breaks might complain about a certain aspect of a computer interface they use on the job, and those sentiments eventually evolve into a proclivity not to use that feature. On the other hand, an explicit influence would be a company policy letter stating to discontinue use of the feature based on data gathered from a formal employee reporting system. Explicit sources might be the only ones easily available to a safety analyst with little experience in the specific domain; SMEs will more readily understand sources of tacit knowledge in the organization.

Examples of Causal Scenarios – Autonomous Cruise Control

Suppose a human driver of a modern car has an option to use an autonomous cruise control (ACC) that maintains a set distance from traffic ahead of it. This feature cannot accurately detect position and velocity of other traffic during inclement weather. A published warning about this exists in the owner's manual, and a small icon that says "ACC-deg" lights up next to the "ACC-on" icon by the speedometer when the ACC is armed but experiencing sensing problems (it will not shut off automatically if this happens). An STPA of the car design begins by referencing top-level hazards, one of which says: "Car violates minimum safe velocity/distance separation to another vehicle on the road." In STPA Step 1, a safety analyst examining the control loop between the human and the ACC (see Figure 1) would generate several UCAs for that hazard, one of which says: "Driver does not disengage ACC during inclement weather."

For STPA Step 2, the analysis sequence in Figure 2 begins at the "Information Set" (data availability). Design feedback listed here includes "ACC-on" and "ACC-deg" icons, speedometer and engine instruments, and visual weather cues. SMEs might offer non-design feedbacks that include engine noise indicative of a struggling ACC. Affordance feedback would include the feel of the button that engages or disengages ACC. An example of a

causal scenario for the UCA would be: “Driver assumes she has disengaged ACC by pushing the ACC button but does not confirm that the ‘ACC-on’ light has extinguished.” A causal scenario informed by the “Observe” section of Figure 2 would be: “‘ACC-deg’ icon not noticed by driver for [specific design reason(s)].”

The process model identifies behavior states, some of which are: velocity, distance to front traffic, and weather condition. Some mode states are: ACC on/off and ACC degraded/not. The analysis also investigates human mode knowledge, to include that ACC becomes brittle in inclement weather (even cloudy days that might otherwise seem safe), and that ACC communicates its status via the icons on the dashboard. An example of a causal scenario would be: “Driver does not know that the ACC-deg icon indicates a system limitation and continues to use ACC.” A further inquiry into “influences” would look at “Rules and Techniques” and identify causal scenarios such as: “Car manual does not sufficiently emphasize the importance of monitoring for an ‘ACC-deg’ icon, even if weather appears normal.” A causal scenario from “Behavioral Standards” would be: “Auto industry does not sufficiently emphasize that drivers of vehicles with autonomy should read the entire warnings section of their manuals.” These scenarios are used by engineers to eliminate them from the system design or to control them. For example, the cruise control software could be redesigned or the driver interface might be improved.

Conclusion

This paper extends the generation of hazardous scenarios in STPA by classifying information useful for investigating human (or intelligent) controller contributions to system hazards. The analysis of the controller was improved by refining several sections, including looking at three levels of the process model that contribute to robust and flexible behavior. Fundamental human considerations were also added in the form of adding new sections covering workspace and variability, and an emphasis was added to differentiate design, non-design, and affordance feedback to the controller. Finally, influence was considered to capture organizational ties to operational behavior.

These techniques add to the already improved ability of STPA to go beyond targeting humans or software for making arbitrary errors. Considerations that contribute to hazardous scenarios have been refined. Ideally, these improvements can be used not only to investigate existing systems but to inform system design, particularly as software and autonomy capabilities improve.

Acknowledgements

The views expressed in this document are those of the authors and do not reflect the official position or policies of the Air Force, Department of Defense, or United States Government. This discussion covers part of doctoral research by the second author. Special thanks to Dr. Cody Fleming, Adam Williams, and Dajiang Suo for collaborating on many concepts that became the methods presented here.

References

Boyd, J. R. (2010). *A discourse on winning and losing* (Lecture notes). Retrieved from http://dnipogo.org/john-r-boyd/

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

Department of Defense (2012). *System safety* (MIL-STD-882). Retrieved from http://everyspec.com/MIL-STD/MIL-STD-0800-0899/MIL-STD-882E_41682/

Leveson, N. (2012). *Engineering a safer world: Systems thinking applied to safety*. MIT Press.

Leveson, N., Pinnel, L. D., Sandys, S. D., Koga, S., & Reese, J. D. (1997). Analyzing software specifications for mode confusion potential. In *Proceedings of a workshop on human error and system development*, Glasgow, Scotland (pp. 132-146).

Rasmussen, J., Pejtersen, A. M., & Goodstein, L. P. (1994). *Cognitive systems engineering*. Wiley.

Stringfellow. M. V. (2010). *Accident analysis and hazard analysis for human and organizational factors* (Doctoral dissertation). Retrieved from http://dspace.mit.edu/handle/1721.1/63224

Thornberry, C. L. (2014). *Extending the human-controller methodology in Systems-Theoretic Process Analysis* (Master’s thesis). Retrieved from http://dspace.mit.edu/handle/1721.1/90801/

Von Bertalanffy, L. (1968). *General system theory: Foundations, development, applications* (Vol. 55). New York: George Braziller.

Wiener, N. (1965). *Cybernetics or control and communication in the animal and the machine* (Vol. 25). MIT Press.