Performing Hazard Analysis on Complex, Software- and Human-Intensive Systems

J. Thomas, S.M.; Massachusetts Institute of Technology; Cambridge, Massachusetts, USA

N. G. Leveson Ph.D.; Massachusetts Institute of Technology; Cambridge, Massachusetts, USA

Keywords: Hazard Analysis, STPA, STAMP

Abstract

STPA (System-Theoretic Process Analysis) is a new, more powerful hazard analysis technique that assumes accidents are caused by inadequate enforcement of constraints on component behavior rather than simply component failures (ref. 3). Accidents in complex systems are often caused by unsafe interactions among components that have not failed. Because STPA includes both component failure accidents and component interaction accidents, it can potentially find more causes of hazards than the older methods, including causes involving software and human errors which usually involve not failures but inadequate or unsafe control actions. The first step in STPA is to identify the unsafe control actions (including failure to adequately handle component failures) that can lead to hazards. The second step is to determine the potential causes of the unsafe control. This paper describes a procedure for identifying potentially unsafe control actions, from which component safety requirements can be derived and more detailed causal analysis performed.

Introduction

New systems are introducing new technology, such as software and computers, and are changing the role of humans in systems from followers of procedures to supervisors of automation and high-level decision makers (refs. 10, 11). In addition, the level of complexity in many of our new systems is leading to accidents where no components may have failed but unsafe interactions among non-failed components lead to the loss. At the same time, traditional hazard analysis techniques assume accidents are caused by component failures or faults (refs. 3, 7) and oversimplify the role of humans (refs. 5, 6). Attempts have been made to extend these techniques for software and more cognitively complex human errors, but the assumptions underlying these old techniques (accidents are caused by component failures or faults) do not match the fundamental nature of many of the systems we are building today. A more powerful model of accident causation is needed and new hazard analysis techniques must be constructed on this extended causality model.

STAMP is a model of accident causation that treats safety as a control problem, rather than as a failure problem (ref 3). While unsafe control includes inadequate handling of failures, it also includes system and software design errors and erroneous human decision making. In STAMP, accidents are viewed as the result of inadequate enforcement of constraints on system behavior. The reason behind the inadequate enforcement may involve classic component failures, but it may also result from unsafe interactions among components operating as designed or from erroneous control actions by software or humans.

Human and automated controllers use a process model (usually called a mental model for humans), which they use to determine what control actions are needed. The process model contains the controller's understanding of 1) the current state of the controlled process, 2) the desired state of the controlled process, and 3) the ways the process can change state. Software and human errors often result from incorrect process models, e.g., the software thinks the spacecraft has landed and shuts off the descent engines. Accidents can therefore occur when an incorrect or incomplete process model causes a controller to provide control actions that are hazardous. While process model flaws are not the only causes of accidents involving software and human errors, it is a major contributor.

STAMP is based on the observation that there are four types of hazardous control actions that need to be eliminated or controlled to prevent accidents:

1) A control action required for safety is not provided or is not followed
2) An unsafe control action is provided that leads to a hazard
3) A potentially safe control action is provided too late, too early, or out of sequence
4) A safe control action is stopped too soon or applied too long

STPA (System Theoretic Process Analysis) is a hazard analysis technique built on STAMP. Identifying the potentially unsafe control actions for the specific system being considered is the first step in STPA. These unsafe control actions are used to create safety requirements and constraints on the behavior of both the system and its components. Additional analysis can then be performed to identify the detailed scenarios leading to the violation of the safety constraints. As in any hazard analysis, these scenarios are then used to control or mitigate the hazards in the system design.

This paper presents a technique that can be used to identify unsafe control actions and demonstrates the technique using an automated door controller system.

Identifying Potentially Unsafe Behavior

Before beginning an STPA hazard analysis, potential accidents and related system-level hazards are identified along with the corresponding system safety constraints that must be controlled. As an illustrative example, consider a simple automated door control system for a train. The accidents to be considered are: injury to a person by falling out of the train, being hit by a closing door, or being trapped inside a train during an emergency. The system-level hazards relevant to this definition of an accident include:

H-1: Doors close on a person in the doorway
H-2: Doors open when the train is not in a station or is not aligned with a station platform
H-3: Passengers/staff are unable to exit during an emergency.

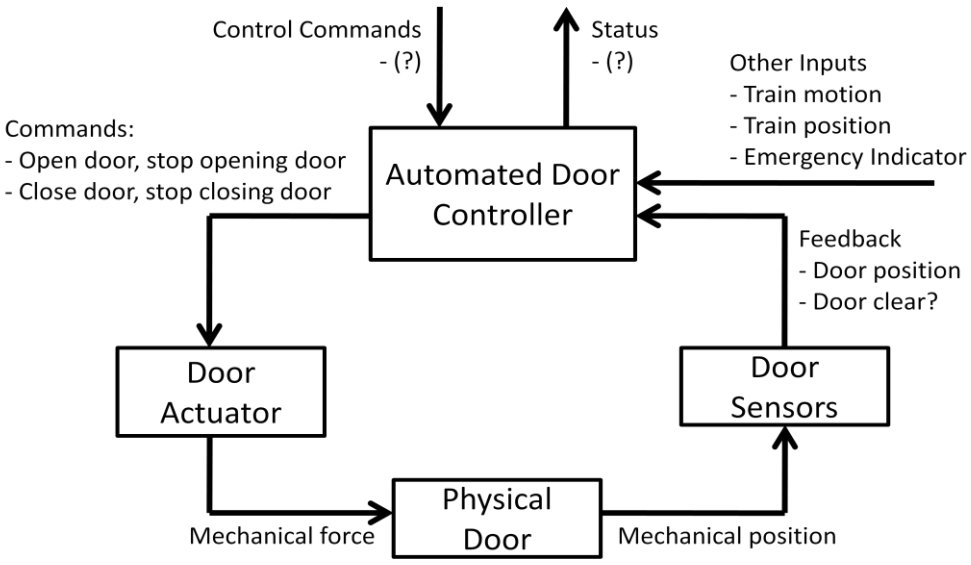

Figure 1: Preliminary control diagram for an automated door controller

STPA is performed on a functional control diagram of the system, which is shown in
Figure 1 for the train door controller. The first part of STPA identifies hazardous control actions for each component that could produce a system-level hazard by violating the system safety constraints. Once the set of hazardous control actions has been identified, the second part of STPA analyzes the system to determine the potential scenarios that could lead to providing a hazardous control action. These scenarios can be used to design controls for the hazards or, if the design already exists, to ensure that these scenarios are adequately controlled.

STPA Step One: The first step of STPA identifies control actions for each component that can lead to one or more of the defined system hazards. The four general types of unsafe control actions were shown above. Hazardous control actions can be documented using a table as in Table 1. The hazardous control actions can then be translated into system and component safety requirements and constraints.

Table 1: Potentially hazardous control actions for a simple automated door controller

Control Action	1) Not Given	2) Given Incorrectly	3) Wrong Timing or Order	4) Stopped too soon or applied too long
Provides door open command	Doors not commanded open once train stops at a platform [not hazardous][1] Doors not commanded open for emergency evacuation [see H-3] Doors not commanded open after closing while a person or obstacle is in the doorway [see H-1]	Doors commanded open while train is in motion [see H-2] Doors commanded open while train is not aligned at a platform [see H-2]	Doors commanded open before train has stopped or after it started moving (same as "while train is in motion") [see H-2] Doors commanded open late, after train has stopped [not hazardous] Doors commanded open late after emergency situation [see H-3]	Door open stopped too soon during normal stop [not hazardous] Door open stopped too soon during emergency stop [see H-3]
Provides door close command	Doors not commanded closed or re-closed before moving [see H-2]	Doors commanded closed while person or object is in the doorway [see H-1] Doors commanded closed during an emergency evacuation [see H-3]	Doors commanded closed too early, before passengers finish entering/exiting [see H-1] Doors commanded closed too late, after train starts moving [see H-2]	Door close stopped too soon, not completely closed [see H-2]

Each item in the table should be evaluated to determine whether it is hazardous as defined by the system-level hazards. For instance, in this simple example the doors remaining closed during a routine train stop (non-emergency) is not hazardous because it does not lead to any of the three hazards specified above. If this situation is a safety concern, then the hazard list can be updated to include the corresponding hazard On the other hand, commanding the doors open while the train is in motion is hazardous because it leads to hazard H-2. Each unsafe control action is then translated into a component-level safety constraint (e.g. train must not be capable of starting with door open, doors must remain closed while train is in motion, etc.).

STPA Step Two: The second step of STPA examines each control loop in the safety control structure to identify potential causal factors for each hazardous control action, i.e., the scenarios for causing a hazard.
Figure 2 shows a generic control loop that can be used to guide this step. While STPA Step One focused on the provided control actions (the upper left corner of
Figure 2), STPA Step Two expands the analysis to consider causal factors along the rest of the control loop.

Consider a hazardous control action for the automated door controller: the doors are commanded closed while a person is in the doorway. STPA Step Two would show that one potential cause of that action is an incorrect belief that the doorway is clear (an incorrect process model). The incorrect process model, in turn, may be the result of inadequate feedback provided by a failed sensor or the feedback may be delayed or corrupted. Alternatively, the designers may have omitted a feedback signal.

[1] This is not hazardous because it does not lead to any of the system-level hazards (see H-1,H-2,H-3 above). If the hazards and accidents included in the safety analysis were extended to include inconvenience to the passengers, then this item would be considered hazardous.

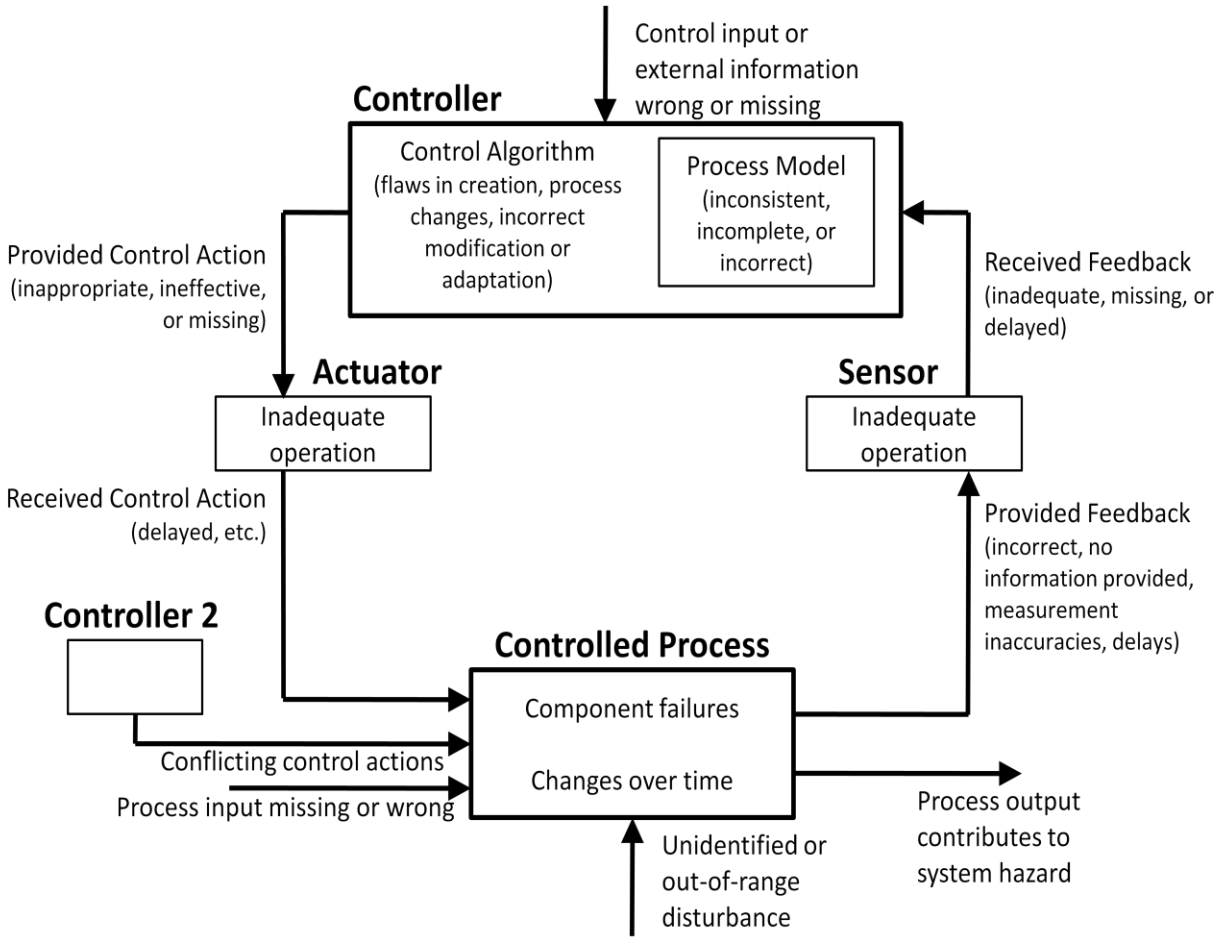

Figure 2: General control loop with causal factors

Once the second step of STPA has been applied to determine potential causes for each hazardous control action identified in STPA Step One, the causes should be eliminated or controlled in the design. STPA has been described in other places [ref. 3] and is not described in detail here due to space limitations.

A Procedure to Identify Hazardous Control Actions

A procedure has been developed to provide additional guidance identifying the hazardous control actions during the first step of STPA. The approach is based on the idea that many control actions are only hazardous in certain contexts. For example, a command to open the doors is not hazardous by itself—it depends on the system state or state of the environment in which the command is given. For example, the command to open train doors is hazardous *when the train is moving*, or *when the train is stopped but misaligned with the platform*. The new procedure involves identifying potential control actions, identifying potentially hazardous states, and then analyzing which combinations together yield a hazardous control action.

Three parts of the procedure are described in the following sections, and each part can be performed independently of the others. The first part deals with control actions that are provided under conditions that make the action hazardous. The second part deals with control actions that are *not* provided under conditions that make inaction hazardous. Finally, the third part considers control actions that are provided in a hazardous way.

Part 1: Control actions provided in a state where the action is hazardous

In this procedure, a controller and the associated control actions are selected from the control structure. In the train example above, the automated door controller can provide four control actions: *open doors*, *stop opening doors*, *close doors*, *or stop closing doors*. Next, the controller's process model is defined to determine the environmental and system states that affect the safety of the control actions.

Controllers use the values of the process model to determine what control actions to provide. In order to make safe decisions, the control algorithm must use process model variable values (i.e., system state or environmental values that are known to the controller). If the controller does not know the values of system state and environmental values that are related to hazards, then the controller cannot be designed to provide safe control actions. Figure 3 shows the required process model for the door controller to carry out its control safely. The required variables in the process model are identified by the definition of the system hazards. For example, hazard H-1 identifies the state of the doorway (whether it is clear or not) as an important environmental variable in deciding whether to close the doors or not.

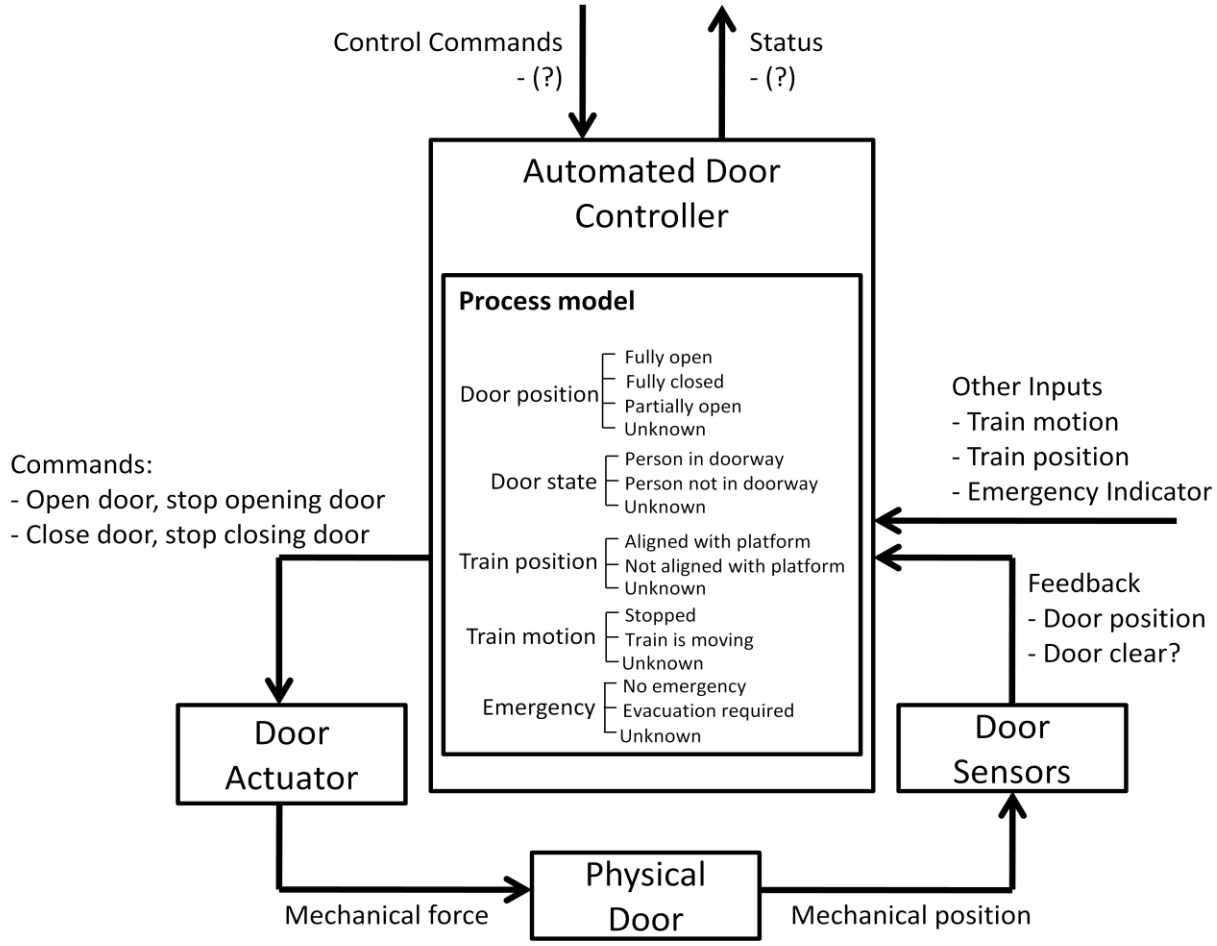

Figure 3: Augmented control structure with the door controller's process model

Once the process model variables have been identified, the potentially hazardous control actions can be identified and changed into safety requirements for the controller. These hazards are identified by examining each potential combination of relevant process model values to determine whether issuing that control action in that state will be hazardous. For example, one possible process model state for the *open door* command consists of the values: the train is stopped, there is no emergency, and the train is not aligned with a platform. Providing the *open door* command in this context is a hazardous control action.

Each row in Table 2 specifies a different context for the ***open door*** command.[2] Context here is defined as a combination of values of the process model variables. Each context is then evaluated to determine whether the control action is hazardous in that context, and the result is recorded in the three columns on the right. The two right-most columns incorporate timing information as well. For example, providing an ***open door*** command in the context of an emergency while the train is stopped is not hazardous; in fact, that's exactly what should happen. However, providing the ***open door*** command *too late* in that context is certainly hazardous.

Table 2: Contexts for the *open door* control action

Control Action	Train Motion	Emergency	Train Position	Hazardous control action?		
				If provided any time in this context	If provided too early in this context	If provided too late in this context
Door open command provided	Train is moving	No emergency	(doesn't matter)	Yes	Yes	Yes
Door open command provided	Train is moving	Emergency exists	(doesn't matter)	Yes*	Yes*	Yes*
Door open command provided	Train is stopped	Emergency exists	(doesn't matter)	No	No	Yes
Door open command provided	Train is stopped	No emergency	Not aligned with platform	Yes	Yes	Yes
Door open command provided	Train is stopped	No emergency	Aligned with platform	No	No	No

*assumption: passengers can exit to the following or proceeding car in an emergency

Note that during this process, some combinations of conditions may expose conflicts in the design that need to be considered. For example, is it hazardous to provide the ***open door*** command during a fire (an emergency) while the train is in motion? In other words, is it safer to keep the doors closed and trap the passengers inside while the train crawls to a complete stop or is it better to open the doors and risk physical injury because the train is moving? These questions can and should prompt exploration outside the automated door controller. For example, that issue might be addressed in the design by providing a way for passengers to exit to nearby train cars when there is an emergency and the train is moving.

Part 2: Control actions not provided in a state that makes inaction hazardous

This part of the procedure considers potential states in which the lack of a control action is hazardous. The same basic process is used: identify the corresponding process model variables and the potential values, create contexts for the action using combinations of values, and then consider whether an absence of the specified control action would be hazardous in the given context. Table 3 shows the hazardous control actions for the door open command not being provided..

[2] Note that a separate table would be constructed for each of the four commands, including the stop commands.

Table 3: Contexts for the lack of an *open door* control action

Control Action	Train Motion	Emergency	Train Position	Door State	Hazardous if not provided in this context?
Door open command not provided	Train is stopped	No emergency	Aligned with platform	Person not in doorway	No[3]
Door open command not provided	Train is stopped	No emergency	Aligned with platform	Person in doorway	Yes
Door open command not provided	Train is stopped	No emergency	Not aligned with platform	(doesn't matter)	No
Door open command not provided	Train is stopped	Emergency exists	(doesn't matter)	(doesn't matter)	Yes
Door open command not provided	Train is moving	(doesn't matter)	(doesn't matter)	(doesn't matter)	No

Part 3: Control actions provided in an unsafe way

While Part 1 and Part 2 consider control actions provided or not provided at different times or conditions, Part 3 considers whether the control action itself can be unsafe even if provided at the right time and under the right conditions. Part 3 applies to complex commands that contain parameters or consist of several lower-level commands, and therefore can be provided in more than one way. For example, an operator may tune a radio at the right time and under the right conditions but enter the wrong frequency, pilots may start to descend at the right time but using an unsafe pitch/thrust setting, or an air traffic controller may provide instructions using incorrect phraseology. In contrast, the simple door controller commands described above do not have any parameters nor do they consist of any lower-level commands, so Parts 1 and 2 are sufficient for identifying the hazardous control actions.

There are often hundreds of different ways in which a complex command might be provided in an incorrect way, but listing every possible variation of a command is not necessary because the same causes usually exist for many different variations. Therefore, for the purpose of providing an input to begin STPA Step Two, a table such as Table 4 can be used to consider more general ways in which the command can be incorrect. We suggest the following three categories as way to identify control actions that may have similar causes, but current research is exploring the potential of alternatives to this approach:

- The complex control action is missing parameters or sub-commands
- The complex control action includes incorrect parameters or sub-commands
- The complex control action includes out-of-sequence parameters or sub-commands

[3] This is not hazardous because it does not lead to any of the system-level hazards (see H-1,H-2,H-3 in the previous section). If the hazards and accidents included in the safety analysis were extended to include inconvenience to the passengers, then this row would describe a hazardous control action.

Table 4: Example of different ways a complex command can be provided in an unsafe way

Control Action	Missing parameters or sub-commands	Incorrect parameters or sub-commands	Out-of-sequence parameters or sub-commands
Robotic arm controller provides command to capture an object	Missing command parameters will result in an incomplete capture [hazardous if object must be captured]	Incorrect command parameters can cause a collision with the object [hazardous if object must not be impacted]	Out-of-sequence parameters will cause the arm to miss the object [hazardous if object must be captured]

Once Parts 1, 2, and 3 are complete, each identified hazardous control action is translated into a safety constraint that must be enforced by the design of the system. STPA Step Two can then proceed as described above to identify the causal factors (scenarios) that can lead to the identified hazardous control actions.

Conclusions:

This paper has described a hazard analysis technique, called STPA, which is based on STAMP. STPA has been applied to many complex systems and has proven to be both feasible and effective in systems for which fault trees had already been created. STPA found the same hazardous scenarios as the fault trees, but also additional ones involving complex software and human errors (ref 12, 13).

This paper also introduced a new procedure for identifying hazardous control actions while performing an STPA hazard analysis. Current research is exploring potential ways in which similar kinds of detailed procedures can be created to assist the analyst during STPA Step Two. Much of the analysis is potentially automatable and we are also exploring this potential.

References

1. Leveson, Nancy. *Safeware: System Safety and Computers*. Reading, MA: Addison-Wesley Publishing Company, 1995.

2. Rasmussen, Jens. "Risk management in a dynamic society: A modeling problem," *Safety Science* 27, No. 2/3 1997: 183-213.

3. Leveson, Nancy. *Engineering a Safer World: Systems Thinking Applied to Safety*. MIT Press, 2011.

4. Tyler, Brian, Crawley, Frank, and Preston, Malcom. *HAZOP: Guide to Best Practice, 2nd Edition*, IChemE, 2008.

5. Dekker, Sydney. *The field guide to understanding human error* Ashgate Publishing, Ltd., 2006.

6. Dekker, Sidney. *Ten Questions About Human Error: A New View of Human Factors and System Safety*, Mahwah, NJ: Lawrence Erlbaum Associate Inc., 2005.

7. Vesely, W., Roberts, N. H. *Fault Tree Handbook*, Government Printing Office, 1987.

8. Woods, D., Branlat, M. *Basic Patterns in How Adaptive Systems Fail*, in Resilience Engineering in Practice: A Guidebook, Ashgate Publishing, Ltd., 2011

9. Stringfellow, M., Thomas, J., Leveson, N. *Considering Humans in Hazard Analysis.* International System Safety Conference, 2009

10. Bainbridge, Lisanne. *Ironies of automation.* In Jens Rasmussen, Keith Duncan, and Jacques Leplat, editors, New Technology and Human Error, John Wiley and Sons, New York, 1987

11. Sarter, Nadine, Woods, David. *How in the world did we ever get into that mode? Mode Error and Awareness in Supervisory Control.* Journal of the Human Factors and Ergonomics Society, Volume 37, Number 1, March 1995

12. Pereira, Steve, Lee, Grady, Howard, Jeffrey. *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, CA, November 2006

13. Ishimatsu, Takuto, Leveson, Nancy, Thomas, John, Katahira, Masa, Miyamoto, Yuko, Nakao, Haruka. *Modeling and Hazard Analysis using STPA.* Proceedings of the International Association for the Advancement of Space Safety Conference, Huntsville, Alabama, May 2010