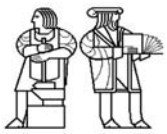


Massachusetts Institute of Technology
Engineering Systems Division

Working Paper Series

ESD-WP-2004-08

MODEL-BASED ANALYSIS OF SOCIO-TECHNICAL RISK[1]

Nancy G. Leveson
Engineering Systems Division
MIT

December 2004

Model-Based Analysis of Socio-Technical Risk[1]

Nancy G. Leveson

Abstract

Traditional approaches to hazard analysis and safety-related risk management are based on an accident model that focuses on failure events in static engineering designs and linear notions of causality. They are therefore limited in their ability to include complex human decision-making, software errors, system accidents (versus component failure accidents), and organizational risk factors in the analysis. These traditional accident models do not adequately capture the dynamic complexity and non-linear interactions that characterize accidents in complex systems, i.e., what Perrow called *system accidents*. System accidents often result from adaptation and degradation of safety over time: The move to a high-risk state occurs without any particular decision to do so but simply as a series of decisions or adaptations (*asynchronous evolution*) that move the system into a high-risk state where almost any slight error or deviation can lead to a major loss.

To handle this more comprehensive view of accidents, risk management tools and models need to treat systems as dynamic processes that are continually adapting to achieve their ends and to react to changes in themselves and their environment. Leveson's new accident model, STAMP (Systems-Theoretic Accident Modeling and Processes), provides the foundation for such a risk management approach by describing the process leading up to an accident as an adaptive feedback function that fails to maintain safety constraints as performance changes over time to meet a complex set of goals and values.

In this report, a new type of hazard analysis based on this new model of accident causation is described called STPA (STAMP-based Analysis). STPA is illustrated by applying it to TCAS II, a complex aircraft collision avoidance system, and to a public water safety system in Canada. In the first example (TCAS II), STPA is used to analyze an existing system design. A formal and executable modeling/specification language called SpecTRM-RL is used to model and simulate the technical and human components in the system and to provide the support required for the STPA analysis. The results are compared with traditional hazard analysis techniques, including a high-quality TCAS II fault tree analysis created by MITRE for the FAA. The STPA analysis was found to be more comprehensive and complete than the fault tree analysis.

The second example of STPA (the public water system) illustrates its application to the organizational and social components of open systems as well as the technical. In this example, STPA is used to drive the design process rather than to evaluate an existing design. Again, SpecTRM-RL models are used to support the analysis, but this time we added system dynamics models. SpecTRM-RL allows us to capture the system's static structure (hardware, software, operational procedures, and management controls) and is useful in performing hazard analyses

[1] This research was partially supported by NASA grants NAG2-1843 and NAS2-03117 and NSF ITR grant CCR-0085829.

that examine complex socio-technical safety control structures. The addition of system dynamics models allows simulation and modeling of the system's behavioral dynamics and the effects of changes over time.

STPA allowed us to examine the impact of organizational decision-making and technical design decisions on system risk and resiliency. The integration of STPA, SpecTRM-RL, and system dynamics creates the potential for a simulation and analysis environment to support and guide the initial technical and operational system design as well as organizational and management policy design. The results of STPA analysis can also be used to support organizational learning and performance monitoring throughout the system's life cycle so that degradation of safety and increases in risk can be detected before a catastrophe results.

Table of Contents

Table of Figures

Introduction

As system complexity is increasing and new types of technology, particularly digital systems and software, are being introduced, hazard analysis and risk management techniques are lagging behind engineering practice. Our long-term research goal is to create commercial model-based simulation and analysis tools for risk management throughout the system life cycle, including design, manufacturing, operations, and maintenance. This report describes a new model-based hazard analysis technique built on a new model of accident causation called STAMP (Systems-Theoretic Accident Modeling and Processes) [Leveson 2003].

Traditional approaches to risk management focus on failure events in static engineering designs. Our approach, in contrast, treats a system as a dynamic process that is continually adapting to achieve its ends and to react to changes in itself and its environment. The original design must not only enforce appropriate constraints on behavior to ensure safe operation, but the system must continue to operate safely as changes occur. To achieve this goal, the process leading up to an accident (loss event) can be described in terms of an adaptive feedback function that fails to maintain safety constraints as performance changes over time to meet a complex set of goals and values.

STPA (STAMP-based Analysis) is a new approach to hazard analysis that enables model-based simulation and analysis of risk throughout the system life cycle, including complex human decision-making, software errors, system accidents (versus component failure accidents), and organizational risk factors.

This report describes STPA. Chapters 1 and 2 provide background on the STAMP model of accidents and the SpecTRM-RL modeling language used in the analysis. They can be skipped by those already familiar with these topics. Chapter 3 shows how STPA can be used to perform hazard analysis on an existing design. TCAS II, a complex aircraft collision avoidance system, is used as the example. TCAS II was chosen because it includes both humans and automation and because a first-rate fault tree analysis already exists for it (created by MITRE for the FAA) that can be used in evaluating STPA. Chapter 4 compares STPA with traditional hazard analysis techniques and evaluates the results of our TCAS II hazard analysis with the information provided in the MITRE fault tree.

Chapter 5 applies STPA to a system with technical, operational, managerial, and government regulatory aspects, i.e. the public water system in Ontario Canada. In this case, rather than performing a post-facto analysis, STPA was used to assist in the design of the public water system, including the safety control structure, and to provide information that would be useful in managing the system and detecting migration toward states of elevated risk during operations. System dynamics models and simulation were added to the basic SpecTRM-RL models to provide a more comprehensive ability to perform dynamic modeling. Finally, conclusions and future research directions are described in Chapter 6.

Chapter 1: The STAMP Model of Accidents

Accident models underlie all efforts to engineer for safety: they form the basis for (1) investigating and analyzing accidents; (2) designing to prevent future losses; and (3) determining whether systems are suitable for use by assessing the risk associated with an activity, the use of a product, or the operation of a system. While most people are not consciously aware that they are using a model when engaged in these activities, some (perhaps subconscious) model of the phenomenon is always part of the process.

Accident models are used to explain how accidents occur. An underlying assumption of all accident models is that there are common patterns in accidents and that they are not simply random events. Accident models impose these patterns on accidents and influence the factors considered in any safety or hazard analysis. Therefore, the model may either act as a filter and bias toward considering only certain events and conditions or it may expand activities by forcing consideration of factors that are often omitted.

The most common accident models used today have their roots in industrial safety and view accidents as resulting from multiple events sequenced as a forward chain over time. The events considered as critical in these models almost always involve some type of component failure or human error, or they are energy related. The chains may be branching (as in fault trees) or may be visualized as holes in Swiss cheese slices [Reason 1990], but all are simply variants on the basic event-chain concept. There may also be multiple chains synchronized by time or common events [Benner 1975].

The causal relationship represented by the chaining is almost always a direct, linear one representing the notion that the preceding event or condition must have been present for the subsequent event to occur, i.e., if event X had not occurred, then the following event Y would not have occurred. As such, event-chain models encourage limited notions of linear causality, and it is difficult or impossible to incorporate non-linear relationships, such as feedback.

In event-based models, the causal factors identified depend on the events that are considered and the selection of the conditions related to these events, i.e., the conditions that link the events. However, other than the physical events immediately preceding or directly involved in the loss, the choice of events to include is subjective and the conditions selected to explain the events is even more so. Event chain models also limit the countermeasures chosen to prevent losses and the quantification of the risk associated with the operation of the system.

Event-based models work best for accidents where one or several physical components fail, leading to a system failure or hazard. However, such models and the hazard analyses based on them can easily miss subtle and complex couplings and interactions among failure events and omit entirely accidents involving no component failure at all. New models that are more effective for managing risks in complex systems will need to account for social and organizational factors, system accidents and dysfunctional interactions, software errors, human errors involving complex decision-making and supervisory control, and system evolution and adaptation over time. Our long-term goal is to create commercially viable, model-based

simulation and analysis tools for risk management that include all these factors and their interactions.

Social and Organizational Factors in Accidents: Event-based accident models are poor at representing systemic accident factors such as structural deficiencies in the organization, management deficiencies, and flaws in the safety culture of the company or industry. The accident model used should encourage a broad view of accident mechanisms that expands hazard analysis beyond the proximate events: A narrow focus on technological components and pure engineering activities may lead to ignoring some of the most important factors in preventing future accidents.

Large-scale engineered systems are more than just a collection of technological artifacts: They are a reflection of the structure, management, procedures, and culture of the engineering organization that created them, and they are also usually a reflection of the society in which they were created. Overlying every technical system is a social system that provides purpose, goals, and decision criteria [Miles 1973]. Effectively preventing accidents in complex systems requires using accident models that include that social system as well as the technology. Risk management needs to take a broad view of accident mechanisms that expands the scope beyond the events that directly precede the physical losses and also includes the purpose, goals, and decision criteria used to construct and operate systems and the organizational, managerial, and regulatory systems that implement these goals and decision criteria.

Several attempts have been made to graft factors other than simple failure events and conditions onto event models, but all have been unsatisfactory, including our own early attempts. The most common approach has been to add hierarchical levels above the event chain. In the seventies, Johnson proposed a model and sequencing method that described accidents as chains of direct events and causal factors arising from contributory factors, which in turn arise from systemic factors (Figure 1) [Johnson 1980].

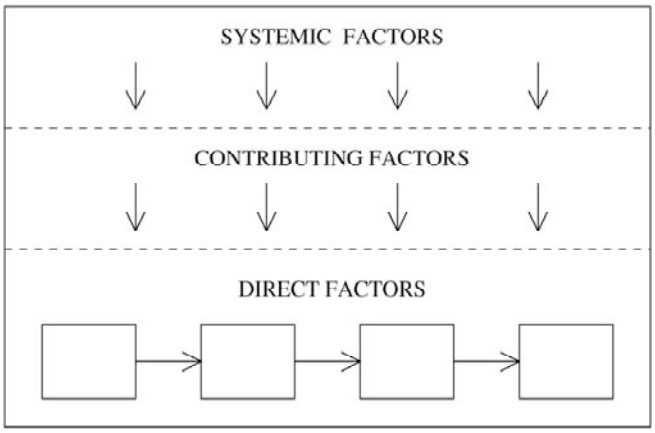


Figure 1 – Johnson's Three Level Model of Accidents

Johnson also tried to add management factors to fault trees in a technique he called MORT (Management Oversight and Risk Tree), but ended up simply providing a general checklist for examining management practices during an accident investigation. While such a checklist can

be very useful, it assumes that every error can be predefined and put into a checklist format. The checklist is comprised of a set of questions that should be asked during an accident investigation. Examples of the questions from a DOE MORT User's Manual include: Was there sufficient training to update and improve needed supervisory skills? Did the supervisors have their own technical staff or access to such individuals? Was the technical support of the right discipline(s) sufficient for the needs of supervisory programs and review functions? Were there established methods for measuring performance that permitted the effectiveness of supervisory programs to be evaluated? Was a maintenance plan provided before startup? Was all relevant information provided to planners and managers? Was it used? Was concern for safety displayed by vigorous, visible personal action by top executives? etc. Johnson originally provided hundreds of such questions, and additions have been made to his checklist since Johnson created it in the 1970s so it is now even larger. The use of the MORT checklist is feasible because the items are so general, but that generality also limits its usefulness, particularly for hazard analysis.

In 1995, Leveson proposed a three-level model (based on an earlier conceptual model by Lewycky) with slightly different types of information [Leveson 1995]. The lowest level describes the accident mechanism in terms of an event chain. For example, an object appeared in front of the car, the driver hit the brakes, the car skidded and hit the tree, the driver was thrown from the car and injured. The second level is a model of the accident in terms of the conditions or lack of conditions that allowed the events at the first level to occur, e.g., the driver does not know how to prevent or stop the skid, the car is not equipped with anti-lock brakes, the driver was driving too fast, the street was wet from rain and thus friction was reduced, visibility was poor, the driver was not wearing a seat belt, the seat belt was defective. Systemic factors make up the third level, i.e., weaknesses of a technical, human, organizational, managerial, or societal nature that contributed to the accident, usually by allowing or causing the conditions to arise that led to the events.

The latest and most sophisticated of these types of hierarchical add-ons to event chains is Rasmussen and Svedung's model of the socio-technical system involved in risk management [Rasmussen 1997, Svedung and Rasmussen 2002]. As shown in Figure 2, at the social and organizational levels they use a hierarchical control structure, with levels for government, regulators and associations, company, management, and staff. At all levels they map information flow. But at each level, they map the factors involved in terms of event chains, with links to the event chains at the level below. In addition they focus on the downstream part of the event chain following the occurrence of the hazard. This downstream approach is common in the process industry, where Rasmussen has done most of his work. In such a downstream focus, emphasis is placed on protection or "safety systems" that identify a hazardous state after it occurs and then attempt to move the plant back into a safe state, often by means of a partial or total shutdown. While this type of shutdown design can work for process plants, it is not appropriate for all types of systems and suffers from a lack of emphasis on designing for safety and eliminating or controlling hazards in the basic system design.

One drawback to all these proposals is that the factors selected to be included are still arbitrary or so general and all-inclusive (such as MORT) as to not be very useful in focusing the analysis on the factors important to the particular system being considered. For a NASA Ames grant, Leveson performed the only experimental evaluation of hierarchical add-ons to event-chain

models of which we are aware. She applied her hierarchical model to eight aerospace accidents and concluded that it was an improvement over a basic chain of events model in terms of accident analysis. Separation of the basic events at Level 1 from all the various types of explanations that could be given for these events at Levels 2 and 3 allowed evaluation of the explanation in a more objective fashion and easier detection of omissions and biases. It also helped to identify conditions indirectly related to events or those related to all or a subset of the events. However, she also found that her model (and the other add-on models that have been proposed) provide little help in selecting the events to include and, more important, in the selection of conditions and systemic factors. She concluded that a different type of model was needed, perhaps one based on system theory.

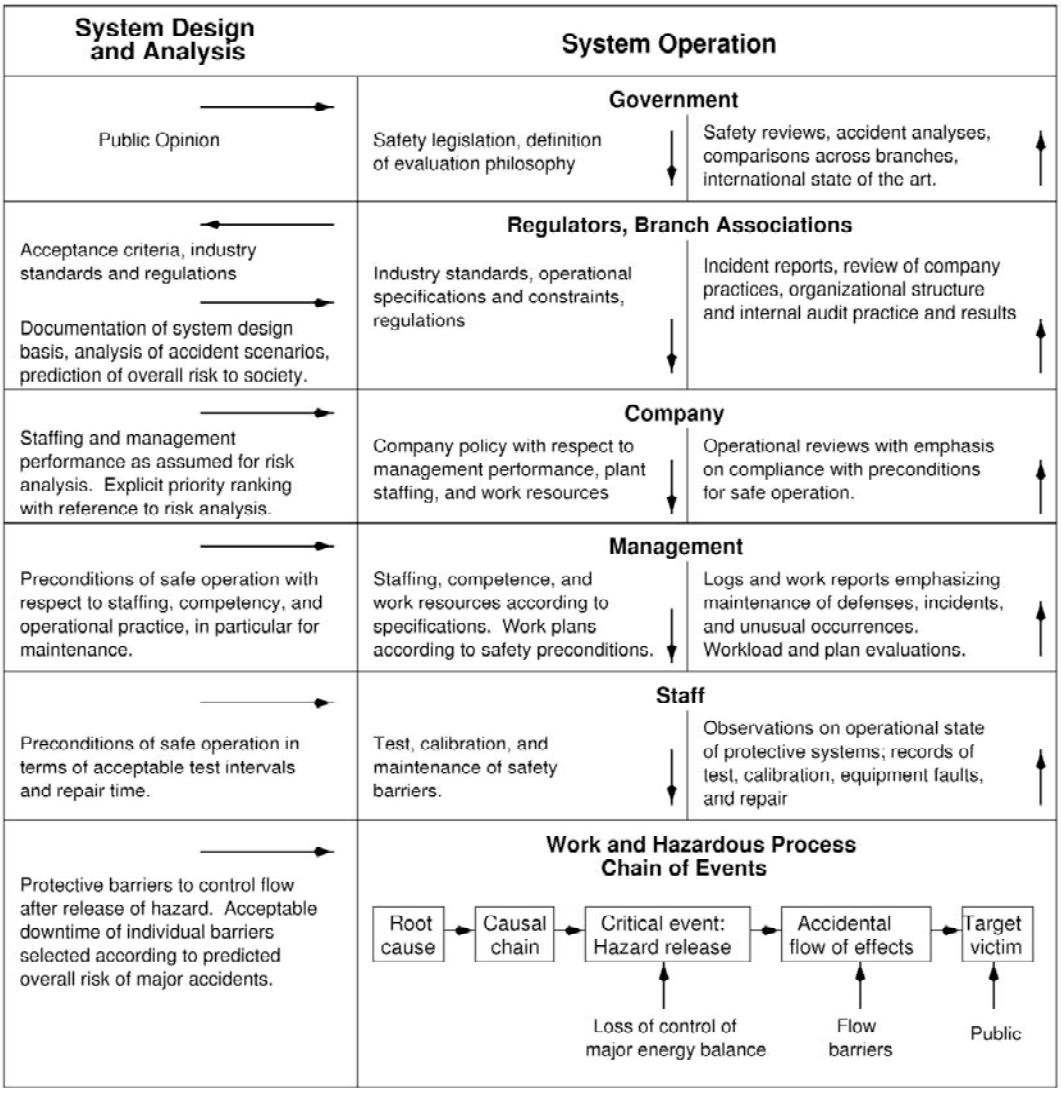


Figure 2 – The Rasmussen/Svedung Model

System Accidents and Software Errors: Since World War II, we have been increasingly experiencing a new type of accident, which Perrow [1984] called a *system accident*, that arises in the interactions *among* system components (electromechanical, digital, and human) rather than in the failure of individual components. In contrast, accidents arising from component failures, including the possibilities of multiple and cascading failures, might be termed *component failure* accidents. Many of NASA's losses in the past few years, such as Mars Polar Lander, were system accidents where all the components operated as specified (i.e., without failure), but there were unplanned interactions among them that led to the loss.

In the simpler systems of the past, analysis and testing allowed exercising the system to detect all potential undesired interactions and changing the system design to eliminate them. Increasing complexity and the introduction of software control is reducing this ability and increasing the incidence of system accidents. System accidents can be explained in terms of inadequate control over component interactions. Prevention requires reducing or eliminating dysfunctional interactions, i.e., interactions that can lead to hazardous states in the controlled process. A taxonomy and classification of the types of dysfunctional interactions leading to accidents based on STAMP is used in the new hazard analysis procedure described in Chapter 3.

While better engineering techniques aimed at increasing component integrity or reducing the effects of component failure on the system are reducing component failure accidents, system accidents are increasing in importance and will require new prevention approaches. In fact, some of the approaches typically used to reduce component failure accidents, such as redundancy, may actually increase the likelihood of system accidents by increasing one of the basic causal factors— interactive complexity. The extraordinary interactive complexity of the systems we are trying to build along with increased coupling between heterogeneous components (which may not be obvious or known) are two of the reasons behind the increase in system accidents. Both are related to the introduction of new technology, particularly software and digital technology.

While system design errors in pure physical systems can usually be detected during system testing and fixed before use, software is allowing us to build systems with such a high level of interactive complexity that potential interactions among components cannot be thoroughly planned, anticipated, tested, or guarded against. We are building systems that are intellectually unmanageable using currently available tools. In the long run, the solution is to extend our ability to manage complexity, but in the present we must deal with the tools we have and catching up (in terms of tool development) to the continually increasing complexity of the systems we would like to build may be difficult.

A second factor implicated in system accidents is tight coupling. Loosely coupled or decoupled components and subsystems allow for intervention when problems arise or they limit the ability of a disturbance in one part of a system to affect other parts. The drive toward tightly coupled systems is fueled by a desire for higher levels of efficiency and functionality. The use of software allows us now to achieve levels of coupling and interaction that were previously impossible with pure electromechanical devices.

Most software-related accidents have been system accidents—they stem from the operation of the software, not from its *lack* of operation and usually that operation is exactly what the software engineers intended. Thus risk management techniques focusing on classic types of failure events will have limited applicability to software.

A more appropriate way to understand the role of software in accidents is to use systems theory. In systems theory terminology, safety is an *emergent property* that arises when the system components interact within an environment. Emergent properties are controlled or enforced by a set of constraints on the system states that result from component interactions; for example, valve A must always be open whenever valve B is closed. Accidents result when safety constraints are violated. When software acts as a controller in such systems, it embodies or enforces the constraints by controlling components and their interactions (e.g., opening and closing the valves). Software, then, can contribute to an accident by not enforcing the appropriate constraints on behavior or by commanding behavior that violates the constraints.

The approach to preventing accidents implied by a systems-based accident model, namely, enforcing constraints on component behavior and interactions, is different than that of approaches that focus on eliminating component failures. Note that accidents caused by basic component failures are also included in the systems accident model. Component failures may result from inadequate constraints on the manufacturing process; inadequate engineering design such as missing or incorrectly implemented fault tolerance; lack of correspondence between individual component capacity (including humans) and the task requirements; inadequate maintenance, including preventive maintenance; etc. A model based on systems theory goes beyond simply blaming component failure for accidents and requires that the reasons be identified for why those failures occurred, why they led to an accident, and what system-level constraints must be imposed to prevent them or prevent hazardous system states if they do occur, potentially leading to more varied and effective measures than simply attempting to handle failures through redundancy.

Human Error: Human tasks and procedures are usually defined using a task analysis in which a task is broken down into a sequence of discrete actions. A human error is then defined as any deviation from the performance of the specified or prescribed sequence. However, human behavior is continually adapting to pressures toward efficiency and other goals. As a result, instructions and written procedures are almost never followed exactly. In fact, a common way for workers to apply pressure to management without actually going out on strike is to *work to rule*, which can lead to a breakdown in productivity and even chaos.

In studies of operators, even in such highly constrained and high-risk environments as nuclear power plants, modification of instructions is repeatedly found and the violation of rules appears to be quite rational, given the actual workload and timing constraints [Fujita 1991, Vicente 1995, Woods 1984]. In these situations, a basic conflict exists between error as seen as a deviation from the *normative procedure* and error as seen as a deviation from the rational and normally used *effective procedure* (Rasmussen 1997).

One implication is that following an accident, it will be easy to find someone involved in the dynamic flow of events that has violated a formal rule by following *established practice* rather

than *specified practice*. Given the frequent deviation of established practice from normative work instructions and rules, it is not surprising that operator "error" is found to be the cause of 70–80% of accidents.

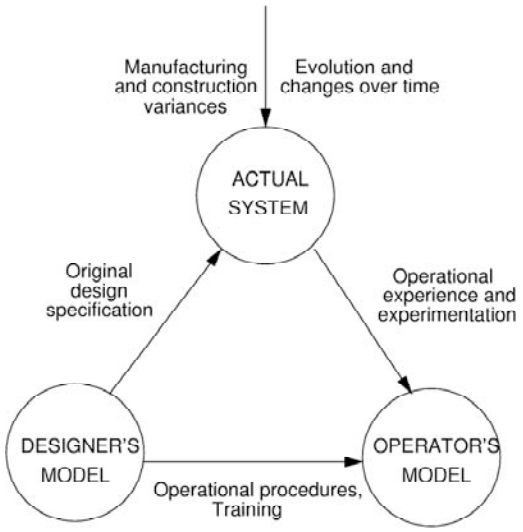


Figure 3 – Mental Models

The updating of human mental models plays a significant role here (Figure 3) and in the hazard analysis approach used in the research supported by this SBIR grant. Both the designer and the operator will have their own mental models of the plant. It is quite natural for the designer's and operator's models to differ and even for both to have significant differences from the actual plant as it exists. During development, the designer evolves a model of the plant to the point where it can be built. The *designer's model* is an idealization formed *before* the plant is constructed. Significant differences may exist between this ideal model and the actual constructed system. Besides construction problems, the designer always deals with ideals or averages, not with the actual components themselves. Thus, a designer may have a model of a valve with an average closure time, while real valves have closure times that fall somewhere along a continuum of behavior that reflects manufacturing and material differences. The designer's model will be the basis for developing operator work instructions and training.

The *operator's model* will be based partly on formal training and partly on experience with the system. The operator must cope with the system as it is constructed and not as it may have been envisioned. In addition, physical systems will change over time and the operator's model must change accordingly. The only way for the operator to determine that the system has changed and that his or her mental model must be updated is through experimentation: To learn where the boundaries of safe behavior currently are, occasionally they must be crossed. The role of such experimentation in accidents cannot be understood by treating human errors as events in a causal chain separate from the feedback loops in which they operate. Actions that are quite rational and important during the search for information and test of hypotheses may appear to be

unacceptable mistakes in hindsight, without access to the many details of a "turbulent" situation [Rasmussen, Pejtersen and Goodstein 1994].

Traditional decision theory research perceives decisions as discrete processes that can be separated from the context and studied as an isolated phenomenon. More recent research has taken a very different approach: Instead of thinking of operations as predefined sequences of actions, human interaction with a system is increasingly being considered to be a continuous control task in which separate decisions or errors are difficult to identify. Edwards, back in 1962, was one of the first to argue that decisions can only be understood as part of an ongoing process. The state of the system is perceived in terms of possible actions, one of these actions is chosen, and the resulting response from the controlled system acts as a background for the next action. Errors then are difficult to localize in the stream of behavior; the effects of less successful actions are a natural part of the search on the part of the operator for optimal performance.

Not only are separate decisions difficult to identify in this model of human control, but the study of decision making then cannot be separated from a simultaneous study of the social context, the value system in which it takes place, and the dynamic work process it is intended to control [Rasmussen 1990]. This view is the foundation of *dynamic decision making* [Brehmer 1992] and the new field of *naturalistic decision making* [Klein *et.al* 1993, Zsambok and Klein 1997].

Using this view of human error leads to a new approach to control of human performance and to risk management: Rather than trying to control behavior by fighting deviations from a particular path, focus should be on control of behavior by identifying the boundaries of safe performance, by making boundaries explicit and known, by giving opportunities to develop coping skills at the boundaries, by designing systems to support safe optimization and adaptation of performance in response to contextual influences and pressures, by designing for error tolerance [Rasmussen 1990], by making errors observable and reversible before safety constraints are violated, and by counteracting the pressures that drive operators and decision makers to violate safety constraints. A goal of our research is to provide automated tools to assist with this approach to dealing with human error in risk management.

Accidents can also result from dysfunctional interactions among decision makers. The safety of individual decisions may depend on the activities of decision makers in other parts of the organization. Accidents then can result from the interaction of the potential side effects of the performance of individual decision makers during their normal work. Most decisions are sound given local judgment criteria and the time and budget pressures and short-term incentives that shape behavior. Experts do their best to meet local conditions and in the busy daily flow of activities are unaware of any potentially dangerous side affects. Each individual decision may appear safe and rational within the context of individual work environments and local pressures, but may be unsafe when considered as a whole: It is difficult if not impossible for any individual to judge the safety of their decisions when that safety is dependent on the decisions made by other people in other departments and organizations. The modeling and simulation tools we plan to create will allow decision makers to evaluate the safety of their decisions within the larger context of the system as a whole.

Adaptation: Any accident model that includes social and organizational factors and human error must account for adaptation. To paraphrase a familiar saying, the only constant is that nothing ever remains constant. Systems and organizations continually experience change as adaptations are made in response to local pressures and short-term productivity and cost goals. People adapt to their environment or they change their environment to better suit their purposes. A corollary of this propensity for systems and people to adapt over time is that safety defenses are likely to degenerate systematically through time, particularly when pressure toward cost-effectiveness and increased productivity is the dominant element in decision making. For example, the redundancy and other precautions added to protect against human error often degenerate over time as work practices adapt to increase efficiency and productivity. The critical factor here is that such adaptation is not a random process—it is an optimization process depending on search strategies—and thus should be predictable and potentially controllable [Rasmussen 1997].

Woods has also stressed the importance of adaptation in accidents. He describes organizational and human failures as breakdowns in adaptations directed at coping with complexity and accidents as a "drift toward failure as planned defenses erode in the face of production pressures and change" [Woods, 2000]. Similarly, Rasmussen has argued that major accidents are often caused not by a coincidence of independent failures but instead reflect a systematic migration of organizational behavior to the boundaries of safe behavior under pressure toward cost effectiveness in an aggressive, competitive environment [Rasmussen 1997].

Humans and organizations can adapt and still maintain safety as long as their behavior conforms to constraints on safe behavior. But in the search for optimal operations, humans and organizations will usually close in on and explore the boundaries of established practice, and such exploration implies the risk of occasionally violating system safety constraints unless those constraints are enforced.

A risk management toolset that handles system adaptation over time must consider the processes involved in accidents and not simply events and conditions: Processes control a sequence of events and describe system and human behavior over time rather than considering events and human actions individually. As Rasmussen argues, deterministic, causal models are inadequate to explain the organizational and social factors in highly adaptive socio-technical systems. Instead, accident causation must be viewed as a complex *process* involving the entire socio-technical system including legislators, government agencies, industry associations and insurance companies, company management, technical and engineering personnel, operations, etc. Tools for hazard analysis and system simulation must include such processes.

STAMP: An Accident Model Based on Systems Theory: A new model of accidents, called STAMP (Systems-Theoretic Accident Model and Processes), has been created upon which the new tools will rest. In STAMP, safety is viewed as a dynamic control problem. The goal is to identify and enforce constraints on system development and operations that result in safe behavior. In this framework, understanding why an accident occurred requires determining why the control structure was ineffective. Preventing future accidents requires designing a control structure that will enforce the necessary constraints.

The most basic concept in the new model is not an event, but a *constraint*. In systems theory terminology, safety is an *emergent property* that arises when the system components interact within an environment. Emergent properties are controlled or enforced by a set of constraints on the system states that result from component interactions; for example, a constraint might be that the descent engines must remain on until the lander reaches the planet's surface. In STAMP, the cause of an accident, instead of being understood in terms of a series of failure events, is viewed as the result of an ineffective enforcement of constraints on the system design and operations. This definition of accidents fits both classic component failure accidents and system accidents. It allows effectively incorporating software, human decision-making, adaptation, and social and managerial factors into the analysis. The goal of risk management using this accident model is not to eliminate component failures but to identify the design constraints necessary to maintain safety and to ensure that the system design, including the social and organizational aspects and not just the physical ones, enforces them.

STAMP goes beyond simply blaming component failure for accidents and requires that the reasons be identified for why those failures might occur, how they could lead to an accident, and what system-level constraints must be imposed to prevent them or to prevent hazardous system states if they do occur. In addition, STAMP includes the prevention of accidents that are not a result of component failure but result from dysfunctional interactions among components, e.g., the spacecraft landing control software interprets noise from a landing leg sensor as an indication the leg has impacted the surface of the planet and turns off the descent engines too early. The STAMP approach to risk management potentially leads to more varied and effective measures than simply attempting to prevent accidents by preventing component failures.

The basis of risk management in STAMP lies in two concepts: (1) hierarchical safety control structures and (2) process models.

Hierarchical Control Structures: In systems theory, systems are viewed as hierarchical structures where each level imposes constraints on the activity of the level beneath it—that is, constraints or lack of constraints at a higher level allow or control lower-level behavior [Checkland 1981]. Safety-related constraints specify those relationships between system variables that constitute the non-hazardous system states; for example, the power must never be on when the access door is open. Control processes that effectively enforce these constraints will limit system behavior to safe changes and adaptations.

Modeling complex organizations or industries in STAMP involves dividing them into hierarchical levels with control processes operating at the interfaces between levels. Figure 4 shows a generic socio-technical control model. Each system, of course, must be modeled to reflect its specific features. The model in Figure 4 has two basic control structures—one for system development (on the left) and one for system operation (on the right)—with interactions between them.

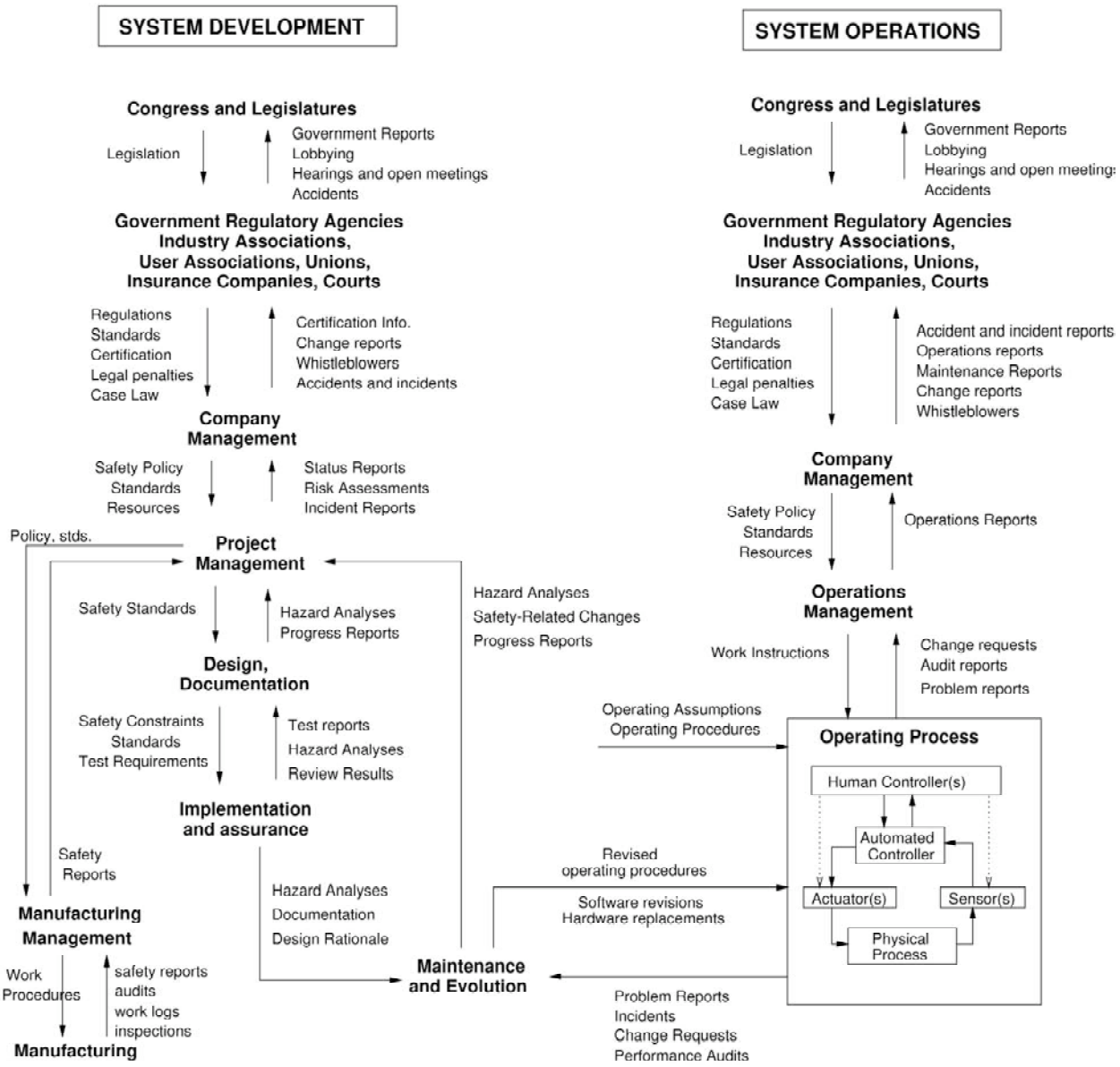


Figure 4 – General Hierarchical Safety Control Structure

At each level of the control structure, inadequate control may result from missing constraints, inadequately communicated constraints, or from constraints that are not enforced correctly at a lower level. Communication is a critical factor here as well as monitoring for changes that may occur and feedback of this information to the higher-level control. For example, the safety analysis process that generates constraints always involves some basic assumptions about the operating environment of the process. When the environment changes such that those assumptions are no longer true (i.e., system adaptation occurs), the controls in place may become inadequate. Embedded pacemakers, for example, were originally assumed to be used only in adults, who would lie quietly in the doctor's office while the pacemaker was being

"programmed." Later they began to be used in children, and the assumptions under which the hazard analysis was conducted and the controls were designed no longer held and needed to be revisited.

In the development control structure (shown on the left in Figure 4), the constraints imposed on behavior by government and other entities must be reflected in the design of company safety policy, standards, and allocation of resources. The company policies and standards are, in turn, usually tailored and augmented by each engineering project to fit the needs of the particular project. New objectives may be added at each level. As an example, while government and/or company standards may require a hazard analysis be performed, the system designers and documenters (including those designing the operational procedures and writing user manuals) may have control over the actual hazard analysis process used to identify specific safety constraints on the design and operation of the system.

The design constraints identified as necessary to control system hazards are passed to the implementers and assurers of the individual system components along with standards and other requirements. Success is determined through test reports, reviews, and various additional hazard analyses. At the end of the development process, the results of the hazard analyses as well as documentation of the safety-related design features and design rationale should be passed on to the maintenance group to be used in the change process.

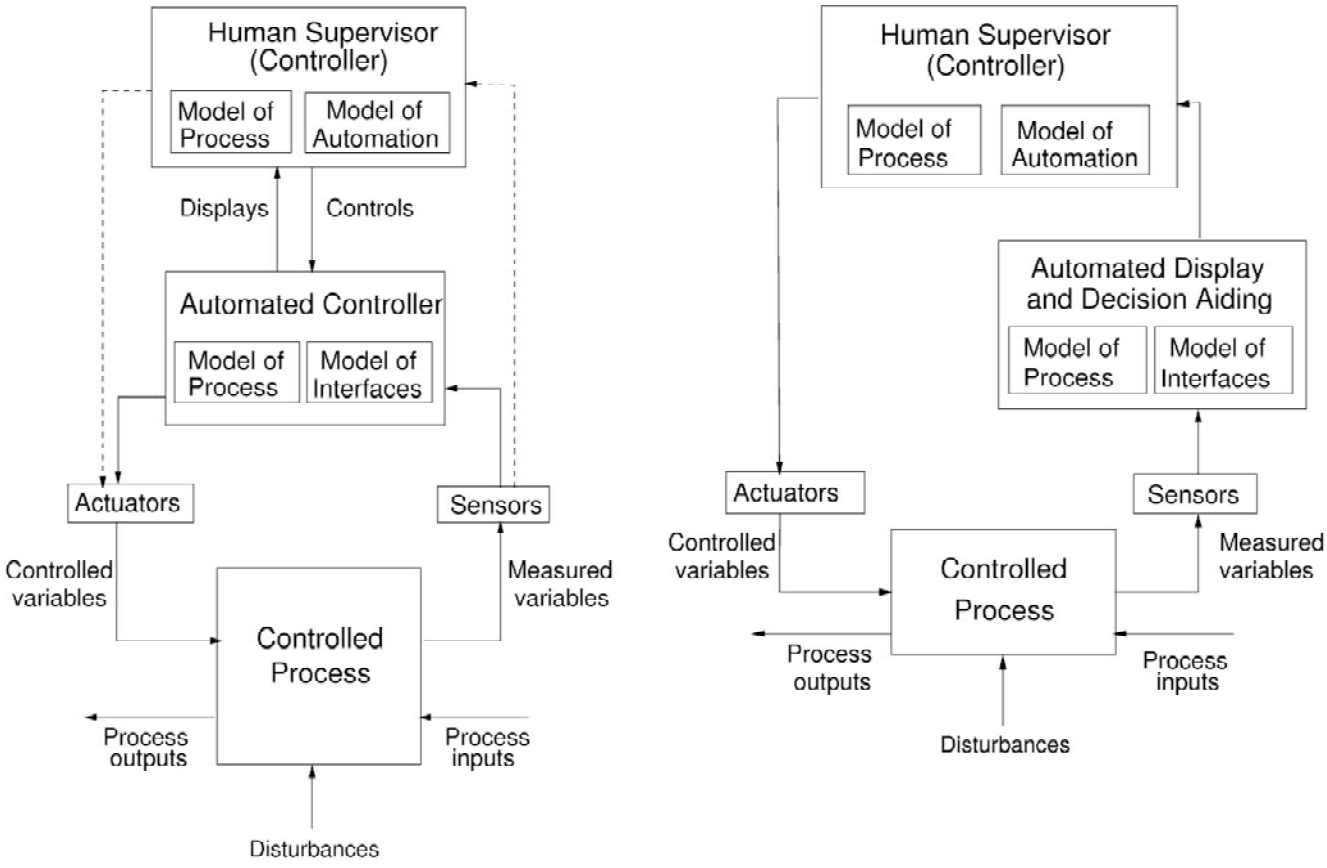
A similar process involving layers of control is found in the system operation control structure (the right half of Figure 4). In addition, there will be (or at least should be) interactions between the two structures. For example, the safety design constraints used during development form the basis for operating procedures and for performance and process auditing.

The safety control structure often changes over time, which accounts for the observation noted above that accidents in complex systems frequently involve a migration of the system toward a state where a small deviation (in the physical system or in human operator behavior) can lead to a catastrophe. The foundation for an accident is often laid years before. One event may trigger the loss, but if that event had not happened, another would have. Union Carbide and the Indian government blamed the Bhopal methyl isocyanate (MIC) release (among the worst industrial accidents in history) on human error—the improper cleaning of a pipe at the chemical plant. However, the maintenance worker was, in fact, only a minor and somewhat irrelevant player in the loss [Leveson 1995]. Instead, degradation in the safety control structure occurred over time and without any particular single decision to do so, but simply as a series of decisions that moved the plant slowly toward a situation where any slight error would lead to a major accident.

Degradation of the safety control structure over time may be related to *asynchronous evolution* [Leplat 1987], where one part of a system changes without the related necessary changes in other parts. Changes to subsystems may be carefully designed, but consideration of their effects on other parts of the system, including the control aspects, may be neglected or inadequate. Asynchronous evolution may also occur when one part of a properly designed system deteriorates. In both these cases, the erroneous expectations of users or system components about the behavior of the changed or degraded subsystem may lead to accidents. The Ariane 5 trajectory changed from that of the Ariane 4, but the inertial reference software did not. One

factor in the loss of contact with the SOHO (SOlar Heliospheric Observatory) spacecraft in 1998 was the failure to communicate to operators that a functional change had been made in a procedure to perform gyro spin-down.

Process Models: A second concept basic to STAMP is that of process models. A control process operates between each level of the hierarchy described above. Figure 5 shows two typical process-control loops, with automation directly issuing control commands in Figure 5a and automation assisting only in human decision making in Figure 5b. To simplify the figures, only one supervisor is shown, but multiple human and automated controllers may be involved.



(a.) Automation directly issuing commands with supervision by a human controller.

(b.) Human control with automated assistance.

Figure 5: Controller Process Models. Each controller contains a model of the process it is controlling.

A basic theorem of systems theory says that any controller must contain a model of the process being controlled to effectively control it [Conant and Ashby 1970]. At one extreme, this process model may contain only one or two variables, such as the model required for a simple thermostat. At the other extreme, effective control may require a complex model with a large

number of state variables and transitions, such as the model needed to control air traffic. SpecTRM-RL (see Chapter 2 and the Appendix) includes this model as a basic part of the process description and therefore provides a basis for analyzing the safety implications of these process models.

Whether the model is embedded in the control logic of an automated controller or in the mental model maintained by a human, it must contain the same type of information: (a) the required relationship among the system variables (the control laws), (b) the current state (the current values of the state variables), and (c) the ways the process can change state.

Human controllers that interact with automated controllers must, in addition to a process model, have a model of the automated device's behavior in order to monitor or supervise it. This lack of understanding of how the automation works is widespread among human controllers and has been implicated in many major accidents [Bureau of Air Safety Investigation 1996]. Later in this report, we show how such models can be used to determine what information is required by operators and managers to allow them to safely operate or manage complex systems.

The process model is used by the human or automation to determine what control actions are needed, and it is updated through various forms of feedback. The same is true for the automation model used by human operators of systems containing automated controllers. Note that a model of the controlled process is required at all levels of the hierarchical control structure, not simply the lowest technical level. Accidents may result from inconsistencies between the process models of the controllers (both human and automated) and the actual process and with each other. The new hazard analysis method described in Chapter 3 directly incorporates this type of inconsistency in the analysis method.

Chapter 2: The SpecTRM-RL Modeling Language

STPA, a new type of hazard analysis based on STAMP, is applied to a control model of the system. While this model may simply be a paper model, the potential exists to create an executable specification/model that forms the basis for a simulation and analysis support environment for the hazard analysis itself and for more general risk management activities. SpecTRM and SpecTRM-RL can be used for this purpose.

SpecTRM (Specification Tools and Requirements Methodology) is a system engineering toolset that focuses on the early stages of system development, where the foundation is set for later implementation, operations, and maintenance activities. The SpecTRM toolset includes support for requirements development and management, hazard analysis, requirements tracing, recording of design rationale, and modeling and analysis of blackbox logic requirements. The formal modeling language, SpecTRM-RL (SpecTRM Requirements Language), is executable and therefore can be used in a simulation environment. In addition, the models are analyzable and tools have been developed to check for completeness, consistency, and robustness. The Appendix contains a description of the SpecTRM-RL language.

SpecTRM has been in development for over 10 years, first only in a university research environment and then, for the past four years, also by Safeware Engineering Corporation. The methodology and tools are currently being used on real industrial projects through Safeware Engineering Corporation contracts with client companies. Because many of the clients were interested in using the tools themselves, a commercial version of the toolset was developed and released for public use on June 15, 2003.

The design of SpecTRM-RL is greatly influenced by the desire to provide a combined specification and modeling language. System specifications (particularly requirements specifications) need to be reviewed and used by people with a large variety of backgrounds and expertise, most of whom are not computer scientists or trained in formal logic or discrete math and may not even be engineers. Therefore, it is not practical to use most formal modeling languages as specification languages. On the other hand, industrial projects rarely have the resources to provide a separate modeling effort for the specification, and the continual changes common to most software development projects will require frequent updates to ensure that the formal model is consistent with the current requirements and system design.

SpecTRM-RL was designed to satisfy both objectives: to be easily readable enough to serve as part of the official specification of the blackbox behavioral requirements and, at the same time, to have an underlying formal model that can be executed and subjected to mathematical analysis.

This underlying formal model based on a Mealy automaton, which we call the *requirements state machine* (RSM), is very low-level and not appropriate as a specification language for complex systems. Instead, SpecTRM-RL acts as the specification language (or visualization of the underlying model) that overlays the low-level model. As long as the mapping from SpecTRM-RL to the RSM is unambiguous and rigorously defined, formal analysis is possible on both the underlying RSM formal model as well as the higher-level SpecTRM-RL specification itself. We

have been experimenting with additional types of visualization to augment the current SpecTRM-RL features [Dulac et.al. 2002].

To assist in readability, we use graphical, tabular, symbolic, and structural notations where we have found each most appropriate for the type of information being specified. Decisions about how things are specified were based on the research literature on visualization, feedback from users of RSML (a previous version of the language) and SpecTRM-RL, our own attempts to build specifications for real systems using the language, observation of the notations engineers use for specifying these properties, and formal experiments on readability of particular types of language features [Zimmerman, Leveson, and Lundqvist 2002].

The SpecTRM-RL notation is driven by the intended use of the language to define a blackbox function from outputs to inputs. Some features are also the result of wanting to remain as close as possible to the way engineers draw and define control loops in order to enhance usability of the language. Other goals for the language design were to eliminate error-prone features, to make the language easy to learn and easy to read and review, and to encourage completeness in specifications.

One goal of the research described in this report was to examine how the SpecTRM-RL language could be extended to model the entire socio-technical system and not just the technical components. Most existing formal specification languages include only the system's technical components, and therefore exclude some of the most important aspects of system design and operations, particularly those related to risk management. A few try to model the human or organizational aspects, but omit the technical. Our goal was to investigate the integration of all these aspects into one modeling and simulation environment to allow integrated system design and risk management. This preliminary research considered only the extensions to SpecTRM-RL necessary to model human and organizational system components. Follow-on research will consider the most appropriate form these extensions might take.

More information on SpecTRM-RL, including some examples, can be found in the Appendix.

Chapter 3: Hazard Analysis using STAMP

A STAMP hazard analysis, or STPA for short, has the same general goals as any hazard analysis: (1) identification of the system hazards and the safety constraints necessary to ensure acceptable risk and (2) accumulation of information about how those constraints could be violated so it can be used in eliminating, reducing, and controlling hazards in the system design and operations.

There are five steps to STPA. The first two are identical to those performed in any system safety analysis. The others either deviate from what is currently done or provide a process for doing what is currently done in an ad hoc manner.

Step 1: Identify the system hazards.

The first step is to identify the system hazards. TCAS II has several hazards, including:

1. TCAS causes or contributes to a near midair collision (NMAC), defined as a pair of controlled aircraft violating minimum separation standards.
2. TCAS causes or contributes to a controlled maneuver into the ground.
3. TCAS causes or contributes to the pilot losing control of the aircraft.
4. TCAS interferes with other safety-related aircraft systems.
5. TCAS interferes with the ground-based Air Traffic Control system (e.g., transponder transmissions to the ground or radar or radio services).
6. TCAS interferes with an ATC advisory that is safety-related, (e.g., avoiding a restricted area or adverse weather conditions) or causes the aircraft to enter an unsafe or restricted part of the airspace.

To limit the size of the example included in this report, only the first hazard is considered. The official MITRE TCAS II fault tree analysis (to which the STAMP-based analysis is compared in the next chapter) included only this NMAC hazard. Four of our six TCAS II system hazards were identified by MITRE as system hazards in the hazard analysis report submitted to the FAA for their risk assessment of TCAS II, but only the NMAC hazard was analyzed.

Step 2: Identify system-level safety-related requirements and constraints.

Once the system hazards are identified, as in any system safety process, the system-level safety-related requirements and constraints must be identified, e.g., "*TCAS must not disrupt the pilot or ATC operations during critical phases of flight nor disrupt routine aircraft operations"* and "*Aircraft trajectory crossing maneuvers must be avoided if possible*." As in any system hazard analysis, these requirements and constraints are derived from examining the potential failures, dysfunctional interactions, or unhandled environmental conditions in the controlled system (the airspace in this case) that could cause the hazard.

STPA starts from these initial system-level requirements and constraints and the set of control actions needed to implement them. As the analysis progresses, new more detailed requirements and constraints will be identified and traced to individual system components

Step 3: Define the basic system control structure.

The next step in the analysis is to define the basic system control structure. Only the operations control structure is considered here (not system development). The operations control structure for TCAS is shown in Figure 6. For this example, it is assumed that the basic system design concept is complete, i.e., each of the system components has high-level functional requirements assigned to it. In fact, STPA can be performed as the system is being defined, and this parallel refinement of the system design and hazard analysis is clearly the best approach. An example is provided in Chapter 5 of this report. Tradeoff decisions and design decisions can then be evaluated for safety as they are being made.

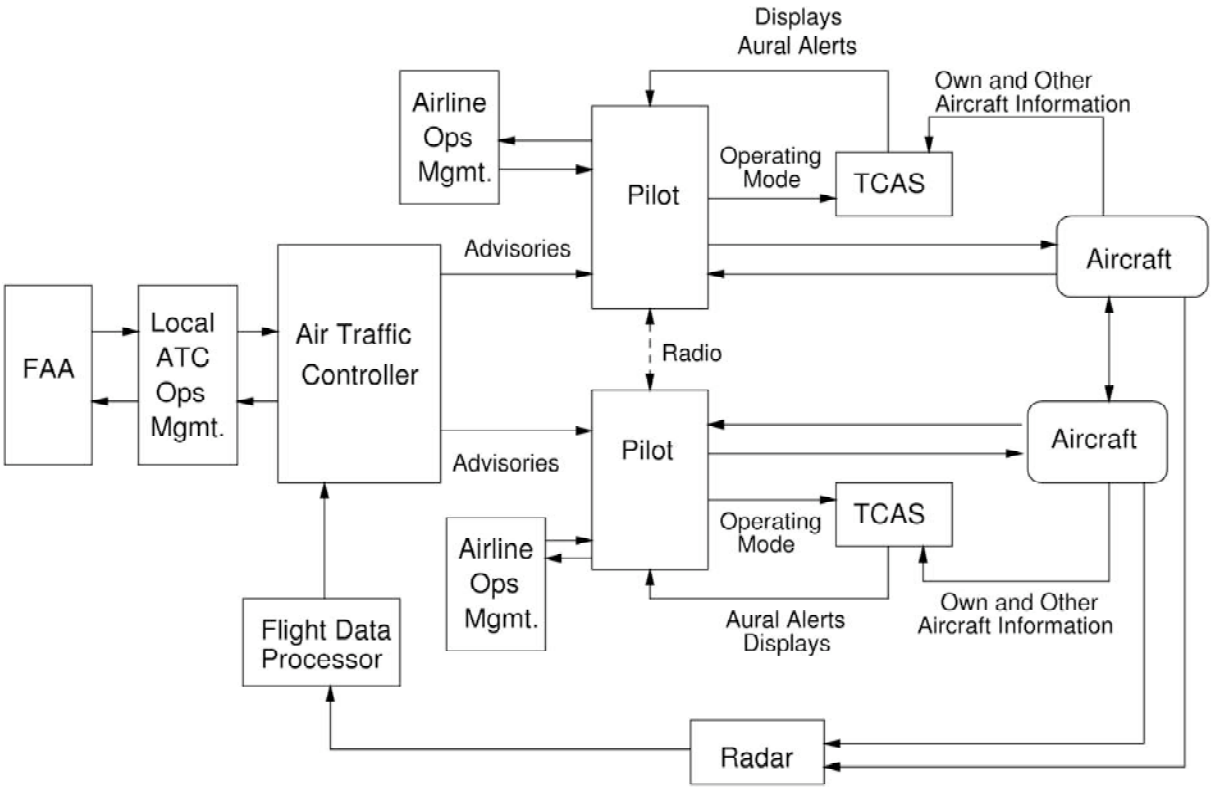


Figure 6 – General TCAS Control Structure

Because our goal was to compare the results of an STPA of TCAS II with the official FAA/MITRE hazard analysis, the current TCAS control structure shown in Figure 6 is assumed. In addition, the management parts of the control structure are not considered in the MITRE hazard analysis, and therefore they are not included in our analysis although they could be (see Chapter 5 for an example). Clearly such an analysis is important for TCAS; for example, there will be constraints that must be enforced by the local air traffic control operations management and airline management such as training pilots and controllers and auditing system performance. The FAA also needs to oversee the entire air traffic control system, including the components involving TCAS and to enforce constraints on how TCAS is developed. The process for generating these management-related constraints is described in Chapter 5.

In the control structure shown in Figure 6, TCAS receives information about its own and other aircraft, analyzes it, and provides the pilot with (1) information about where other aircraft in the vicinity are located and (2) an escape maneuver to avoid potential NMAC threats. The pilot is responsible for implementing the TCAS escape maneuvers and avoiding other aircraft. The ground-based air traffic controller is responsible for maintaining separation between aircraft in the controlled airspace and providing advisories (control actions) for the pilot to follow.

Step *4: Identify inadequate control actions that could lead to a hazardous system state.*

After the general control structure has been defined (or a candidate structure has been identified), the next step is to determine how the controlled system (the two aircraft) can get into a hazardous state. STAMP assumes that hazardous states (states that violate the safety constraints) are the result of ineffective control. Therefore, STPA starts by identifying these potentially inadequate control actions. A controller can provide four general types of inadequate control:

1. A required control action is not provided.
2. An incorrect or unsafe control action is provided.
3. A potentially correct or adequate control action is provided too late (at the wrong time).
4. A correct control action is stopped too soon.

Control actions may be required to handle component failures, environmental disturbances, or dysfunctional interactions among the components. Incorrect or unsafe control actions may also *cause* dysfunctional behavior or interactions among components.

Control actions in TCAS are called *resolution advisories* or RAs. An RA is an aircraft escape maneuver created by TCAS for the pilots to follow. Example resolution advisories are *DESCEND, INCREASE RATE OF CLIMB TO 2500 FPM,* and *DON'T DESCEND*. For the TCAS component of the control structure in Figure 3 and the NMAC hazard, the four types of control flaws translate into:

1. The aircraft are on a near collision course and TCAS does not provide an RA that avoids it.
2. The aircraft are in close proximity and TCAS provides an RA that degrades vertical separation.
3. The aircraft are on a near collision course and TCAS provides a maneuver too late to avoid an NMAC.
4. TCAS removes an RA too soon.

These inadequate control actions can be restated as constraints on the behavior of TCAS:

1. TCAS must provide resolution advisories that avoid near mid-air collisions.
2. TCAS must not provide resolution advisories that degrade vertical separation between two aircraft.
3. TCAS must provide the resolution advisory while enough time remains for the pilot avoid an NMAC. (A human factors analysis should be performed at this point to determine exactly how much time that implies.)
4. TCAS must not remove the resolution advisory before the NMAC is resolved.

For the pilot, the inadequate control actions are:

1. The pilot does not provide a control action to avoid a near mid-air collision.
2. The pilot provides a control action that does not avoid the NMAC.
3. The pilot provides a control action that causes an NMAC that would not otherwise have occurred.
4. The pilot provides a control action that could have avoided the NMAC but was too late.
5. The pilot starts a control action to avoid an NMAC but stops it too soon.

Again, these inadequate control actions can be restated as safety constraints. Similar hazardous control actions are identified for each of the other system components.

Step 5: Determine how these constraints could be violated and attempt to eliminate them, or if that is not possible, to prevent or control them in the system or component design.

At this point, we can start using SpecTRM-RL to provide models for the analysis. A continuous simulation environment can be maintained as the system design progresses and the effects of design decisions can be evaluated (both through simulation and analysis) as they are proposed.

Human modeling with SpecTRM-RL adds complications over the use of the language to model the blackbox input-output behavior of hardware or software. Clearly, humans do not necessarily follow the normative procedures assumed by the system designers. STPA starts with the normative procedures (to ensure that they are at least safe) but then human factors experts and system dynamics models (see Chapter 5), can be used to evaluate what types of deviations are likely and their impact on the hazard. Alternatively, the analysis can start from the hazard and determine what types of deviations would lead to it and attempt either to change the design to eliminate the problem or design the system to prevent or reduce the likelihood of the deviation. For humans, the mitigation strategies may include training, auditing procedures to detect when the deviations occur during operations, etc. This process is no different than that used for hardware or software except that the "failure" mechanism is different, i.e., the hardware may not perform the required function due to physical degradation and the software may not correctly implement its requirements.

For hardware, software, and human components of the system, the information provided by the hazard analysis can be used (1) to guide the test and verification procedures (or training for humans), (2) to change the overall system design to provide protection against the error, or (3) to add fault tolerant features to the component itself to protect against the identified hazardous

behavior. The hazard analysis process, particularly the next steps, not only provides the safety requirements and constraints for the system component, but it provides guidance for this fault tolerant design process.

In this step of the hazard analysis, the analyst determines how the potentially hazardous control actions can occur (the safety constraints can be violated) and either eliminates them through system or component design or controls or mitigates them in the design or in operations. As the system design progresses, more ways to get into the hazardous state will be identified. The analysis can start early (and should in order to have a chance to eliminate hazards), but it will need to be continued and augmented throughout the system design and development process.

At this point, enough information is available to eliminate through overall system redesign or controls some of the inadequate control actions that have been identified. For example, consider the constraint that the pilot must not stop the RA maneuver too soon (before the encounter is resolved). One alternative for the system designer to consider is to take the responsibility for implementing the RA away from the pilot and automating it. While this approach does prevent the inadequate control action on the part of the pilot, it complicates the software and simply substitutes a new potentially inadequate control action on the part of the software. These decisions about allocation of functionality (and in this case, redesign of the TCAS control structure shown in Figure 6) need to be considered as the hazard analysis (and system design) continues. Because this example analysis takes TCAS as it exists today, we pursue this line of reasoning and analysis no further here. The process for an integrated system design/hazard analysis is described in Chapter 5.

Step 5a: Create the process models for the system components.

The basic control structure in Figure 6 is first augmented with the process (plant) model required for each of the control components in the control structure. Additions may be needed as hazards are identified and analyzed and mitigations proposed. An important aspect of STPA is that the use of formal models allows the mitigation features and the impact of various types of faults in other components of the control structure to be evaluated during the hazard analysis process. If the hazard analysis is performed before the system is constructed (as it should be), at the end of the STPA all the safety-related system and component requirements and constraints will have been identified.

As an example, Figure 7 shows the information TCAS needs to know to create RAs. This process model contains information about the controlled aircraft (altitude, speed, bearing, etc.), information about other aircraft that are potential threats (ranges, bearings, altitudes, etc.), and models of the current state of the pilot displays and controls. Some of the information contained in the process model will be obtained through direct inputs from sensors while other information will be derived from processing those inputs (e.g., the threat status of the other aircraft). SpecTRM-RL AND/OR tables describe the logic for updating and using the process variable values.

The process (internal models) of each of the other parts of the TCAS control structure must also be modeled. For example, the pilots must have a model of their own aircraft state, information

about other aircraft that are threats or potential threats, and a model of the operational state of their TCAS systems and any current resolution and traffic advisories The ATC controller must have a model of the critical airspace features (e.g., the location of aircraft and predicted trajectories and conflicts).

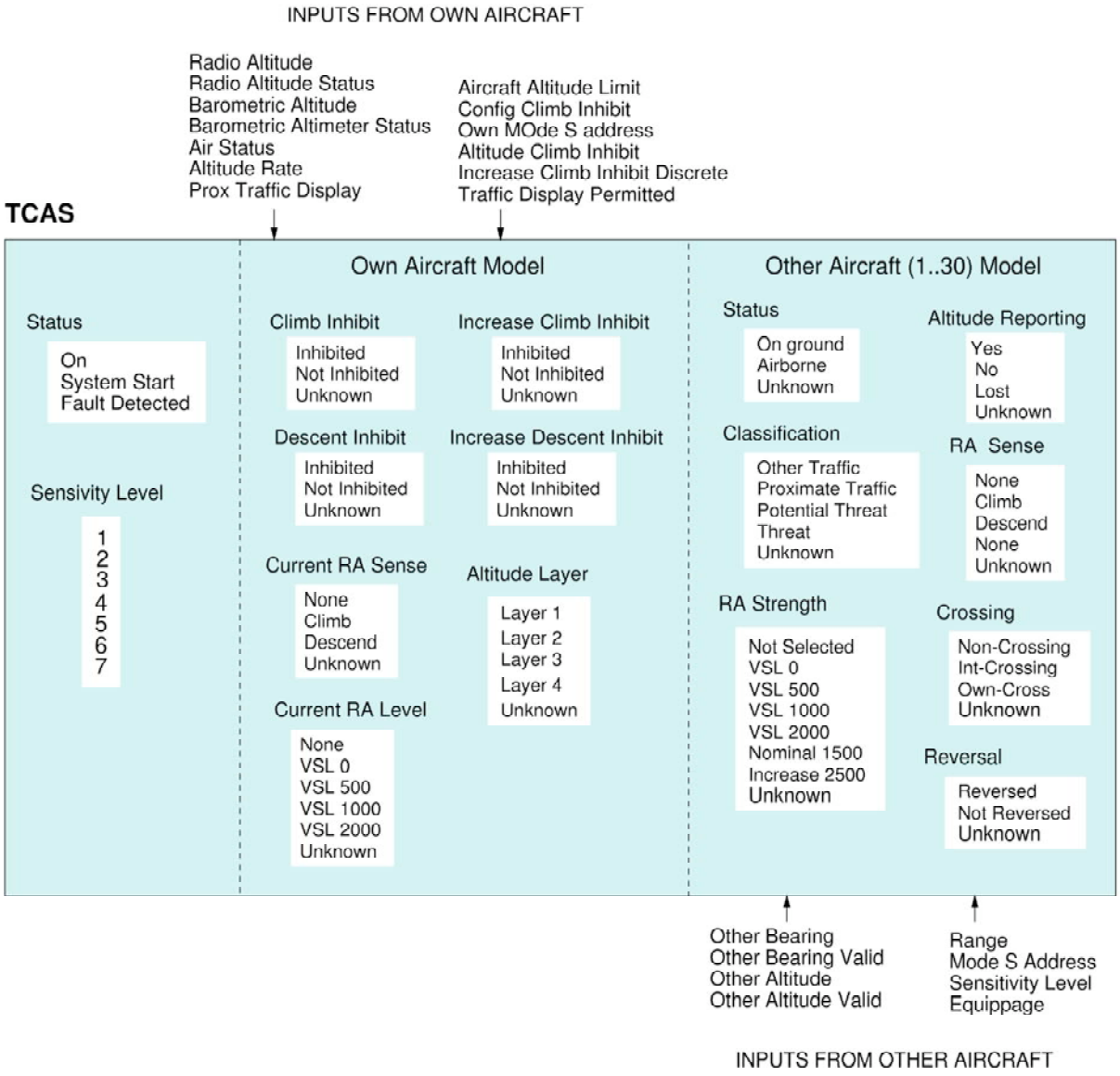


Figure 7 – Process Model for the TCAS Component

Step 5b: For each of the inadequate control actions identified, the parts of the control loop within which the controller is embedded are examined to determine if they could cause or contribute to the inadequate control action (or lack of a necessary control action).

STPA uses the formal SpecTRM-RL model of the system and a set of generic control loop flaws. If we conceive of accidents as resulting from inadequate control and enforcement of safety constraints, then the process that leads to accidents can be understood in terms of flaws in the system development and system operations control structures in place during design, implementation, manufacturing, and operation. These flaws can be classified and used during accident analysis to assist in identifying all the factors involved in the accident or during hazard analysis and other accident prevention activities to identify required constraints. Figure 8 shows the general classification.

1. **Inadequate Enforcement of Constraints (Control Actions)**
   - 1.1 Unidentified hazards
   - 1.2 Inappropriate, ineffective, or missing control actions for identified hazards
     - 1.2.1 Design of control algorithm (process) does not enforce constraints
       - Flaw(s) in creation process
       - Process changes without appropriate change in control algorithm (asynchronous evolution)
       - Incorrect modification or adaptation
     - 1.2.2 Process models inconsistent, incomplete, or incorrect (lack of linkup)
       - Flaw(s) in creation process
       - Flaws(s) in updating process (asynchronous evolution)
       - Time lags and measurement inaccuracies not accounted for
     - 1.2.3 Inadequate coordination among controllers and decision makers (boundary and overlap areas)
2. **Inadequate Execution of Control Action**
   - 2.1 Communication flaw
   - 2.2 Inadequate actuator operation
   - 2.3 Time lag
3. **Inadequate or missing feedback**
   - 3.1 Not provided in system design
   - 3.2 Communication flaw
   - 3.3 Time lag
   - 3.4 Inadequate sensor operation (incorrect or no information provided)

Figure 8 - A Classification of Control Flaws Leading to Hazards

In each control loop at each level of the socio-technical control structure, unsafe behavior results from either a missing or inadequate constraint on the process at the lower level or inadequate enforcement of the constraint leading to its violation. Because each component of the control loop may contribute to inadequate control, classification starts by examining each of the general control loop components and evaluating their potential contribution: (1) the controller may issue inadequate or inappropriate control actions, including inadequate handling of failures or

disturbances in the physical process; (2) control actions may be inadequately executed, or (3) there may be missing or inadequate feedback. These same general factors apply at each level of the socio-technical safety control structure, but the interpretations (applications) of the factor at each level may differ. For all of the factors, at any point in the control loop where a human or organization is involved, it will be necessary to evaluate the context in which decisions are made and the behavior-shaping mechanisms (influences) at play in order to understand the types and reasons for potentially unsafe decisions to be made and to design controls or mitigation measures for them.

Inadequate Enforcement of Safety Constraints: The first factor, inadequate control over (enforcement of) safety constraints, can occur either because hazards (and therefore constraints) were not identified (1.1 in Figure 8) or because the control actions do not adequately enforce the constraints (1.2). The latter may, in turn, result from flawed control algorithms (1.2.1), inconsistent or incorrect process models used by the control algorithms (1.2.2), or by inadequate coordination among multiple controllers and decision makers (1.2.3).

Inadequate Control Algorithms: Control algorithms may not enforce safety constraints (1.2.1) because they are inadequately designed originally, the process may change and thus they become inadequate, or they may be inadequately modified by maintainers (if they are automated) or through various types of natural adaptation if humans implement them.

The simulation and analysis tools for SpecTRM-RL models can be used to check whether control algorithms enforce the safety constraints. The readability of the models also allows for human review. In addition, the algorithms can easily be changed in the AND/OR tables to evaluate the impact on the system and to evaluate alternative algorithms. Consider the unsafe behavior of TCAS not providing an RA when required to avoid an NMAC. Examination of the logic (AND/OR table) for determining when an RA is produced would find the condition that neither aircraft must be on the ground. That would lead to an examination of the logic for determining when an aircraft is on the ground, and so on. Having a readable model should greatly assist the analyst in performing the hazard analysis.

Once an effective and safe control structure has been designed, the analysis must also consider how the designed controls could degrade in effectiveness over time. Many accidents relate to *asynchronous evolution* where one part of a system (in our case the hierarchical control structure) changes or deteriorates without the related necessary changes in other parts. Changes to subsystems may be carefully designed, but consideration of their effects on other parts of the system, including the control aspects, may be neglected or inadequate. The erroneous expectations of users or system components about the behavior of the changed or degraded subsystem may lead to accidents.

Hazard analysis must consider such asynchronous evolution of the safety control structure over time. STAMP analyses use systems dynamics models (see Chapter 5) to model dynamic behavior. These dynamic models can be used to identify likely changes over time that are detrimental to safety. This information can be used to design protection into the system design including operational procedures, to establish controls over changes and maintenance activities, and to design auditing procedures and performance measures (metrics) and management

feedback channels to detect unsafe changes that occur during operations. Note the usefulness of this information in the design of automated information collection during operations and the design of system health management systems. In addition, the information is important in guiding maintenance and upgrade activities and analyzing potential changes to the system or the impact of changes in the environment. The FAA for the past 10 years has been using the executable RSML (our predecessor language to SpecTRM-RL) model of TCAS II created by Professor Leveson and her students to evaluate all changes and upgrades to TCAS II before they are implemented.

Inconsistent Process Models: Accidents, particularly system accidents, often result from inconsistencies between the model of the process used by the controllers (both human and automated) and the actual process state (1.2.2): for example, TCAS may think the aircraft is on the ground and not provide a resolution advisory for a potential threat aircraft or TCAS may have an incorrect model of where the other aircraft is located and its bearing and heading. The situation becomes more even complicated when there are multiple controllers (both human and automated) because each of their process models must also be kept consistent.

How do the models become inconsistent? First, they may be wrong from the beginning (e.g., incorrect software requirements). In this case, the design of the controller itself is flawed: there may be uncontrolled disturbances, unhandled process states, inadvertent commands of the system into a hazardous state, unhandled or incorrectly handled system component failures, etc.

In addition to not starting with an accurate model, models may become incorrect due to lack of feedback, inaccurate feedback, or inadequate processing of the feedback. A contributing factor cited in the Cali B-757 accident report was the omission of the waypoints behind the aircraft from cockpit displays, which contributed to the crew not realizing that the waypoint for which they were searching was behind them (missing feedback to the pilot). The model of the Ariane 501 attitude used by the attitude control software became inconsistent with the launcher attitude when an error message sent by the inertial reference system was interpreted by the attitude control system as data (incorrect processing of feedback), leading to the issuance of an incorrect and unsafe control command.

Consider again the possibility of TCAS not providing an RA when one is required to avoid an NMAC. One way this omission might occur is if the model of the other aircraft is not consistent with the other aircraft's actual state, i.e., the model does not classify the other aircraft as a potential threat. How could this occur? Process models, as stated above, have three parts: initial values for the process variables, a current state, and an algorithm for updating the current state over time based on inputs from sensors and other sources. Each of these must be examined for their potential to cause the hazardous control action. Critical information can be identified in this way, for example, the information used to determine whether an aircraft is on the ground (and thus an RA will not be generated). This information can be used in the design of exception-handling and fault tolerance.

Accidents often occur during initial system startup and at restart after a temporary shutdown, and particular attention needs to be paid to the consistency of the controller's inferred process model and the state of the controlled process at these times. For example, the TCAS logic assumes the

system will be powered up while the aircraft is on the ground. This assumption is important because no RAs are provided when the aircraft is below 500 feet above ground level in order to avoid interfering with landing operations. If TCAS is powered up in the air (perhaps when the pilot discovers a possible problem and reboots the system) and the initial default value for altitude is zero, no RAs will be provided until TCAS gets a reading from its own altimeter even though the pilot may think that TCAS is fully operational. Because this type of initialization problem upon startup is so commonplace, SpecTRM-RL includes (and encourages the use of) an *"unknown"* value for every state variable in the process models. *Unknown* is used as the default value at startup or after loss of critical input sensors until synchronization of the system and its environment is reestablished.

Another reason for not issuing an RA when required (and for the other types of inadequate control actions) might be related to times lags in the control loop or to inaccuracies in the measurements used to derive the current values of state variables in the process model. For example, in TCAS II, relative range positions of other aircraft are computed based on round-trip message propagation time. Measurement inaccuracies and time lags can affect that computation and must be considered in the hazard analysis.

Information about the process state has to be inferred from measurements. The theoretical control function (control law) uses the true values of the controlled variables or component states (e.g., true aircraft positions). The controller, however, must use measured values to infer the true conditions in the process and, if necessary, to derive corrective actions to maintain the required process state. In TCAS, sensors include devices such as altimeters that provide measured altitude, but not necessarily true altitude. The mapping between measured or assumed values may be flawed or time lags that are not accounted for may cause the process model to be incorrect.

Control actions will, in general, lag in their effects on the process because of delays in signal propagation around the control loop: an actuator may not respond immediately to an external command signal (called *dead time*), the process may have delays in responding to manipulated variables (*time constants*), and the sensors may obtain values only at certain sampling intervals (*feedback delays*).

Time lags restrict the speed and extent with which the effects of disturbances, both within the process itself and externally derived, can be reduced. Time lags may also occur between issuing a command and the actual process state change, such as pilot response delays and aircraft performance limitations (which will affect the aircraft trajectory). Finally, time lags may impose extra requirements on the controller, for example, the need to infer delays that are not directly observable. Depending on where in the feedback loop the delay occurs, different process models and controls may be required to cope with the delays: dead time and time constants require a model that makes it possible to predict when an action is needed before the need arises while feedback delays require a model that allows prediction of when a given action has taken effect and when resources will be available again.

Inadequate Coordination Among Controllers or Decision Makers: System accidents often result from inadequate coordination among multiple controllers (human and/or automated) and

other decision makers (1.2.3), including unexpected side effects of decisions or actions or conflicting control actions. Communication flaws play an important role here. Accidents are most likely in *boundary areas* or in *overlap areas* where two or more controllers (human and/or automated) control the same process [Leplat 1987]. In both boundary and overlap areas, the potential for ambiguity and for conflicts among independently made decisions exists (Figure 9).

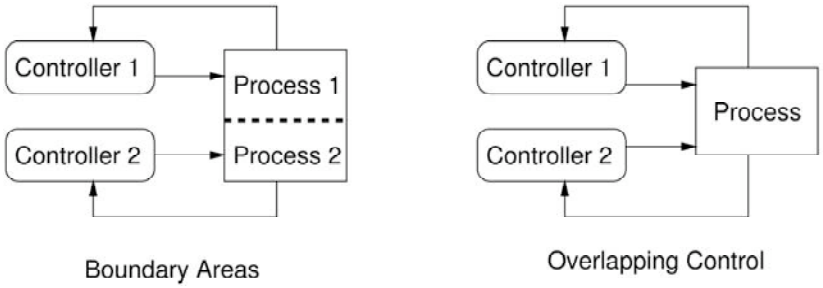


Figure 9 – Two Types of Designs with Potential for Coordination Problems

When controlling boundary areas, there can be confusion over who is actually in control (which control loop is currently exercising control over the process), leading to missing control actions. The functions in the boundary areas are often poorly defined. This type of dysfunction can be related to the number of management levels separating the workers in the departments from a common manager: The greater the distance, the more difficult the communication, and thus the greater the uncertainty and risk.

Examples of accidents involving coordination problems in the control over boundary areas are rife. A Milstar satellite was lost due to inadequate attitude control of the Titan/Centaur launch vehicle, which used an incorrect process model based on erroneous inputs in a software load tape. After the accident, it was discovered that nobody had tested the software using the actual load tape—everyone assumed someone else was doing so. In this case, system engineering and mission assurance activities were missing or ineffective, and a common control or management function was quite distant from the individual development and assurance groups. A factor in the loss of the Black Hawk helicopters to friendly fire over northern Iraq was that the helicopters usually flew in the boundary areas of the No-Fly-Zone, and procedures for handling aircraft in those areas were ill defined. Another factor was that an Army base controlled the flights of the Black Hawks while an Air Force base controlled all the other components of the airspace: A common control point again was high above where the accident occurred in the control structure, and communication problems between the Army and Air Force exacerbated the problems.

Overlap areas exist when a function is achieved by the cooperation of two controllers or when two controllers exert influence on the same object. Such overlap creates the potential for conflicting control actions (dysfunctional interactions among control actions). In an A320 accident in Bangalore, India, for example, the pilot had disconnected his flight director during approach and assumed that the co-pilot would do the same. The result would have been a mode configuration in which airspeed is automatically controlled by the autothrottle (the *speed* mode), which is the recommended procedure for the approach phase. However, the co-pilot had not

turned off his flight director, which meant that *open descent* mode became active when a lower altitude was selected instead of *speed* mode, eventually contributing to the crash of the aircraft short of the runway.

In both boundary and overlap areas, the potential for ambiguity and for conflicts among independently made decisions exists, and the implications of having multiple controllers or multiple sources of control actions must be examined. In the TCAS control structure shown in Figure 3, we see that both the ground air traffic controller and TCAS provide advisories to the pilot. If these advisories conflict (e.g., the ground ATC provides an advisory that conflicts with the advisory provided by TCAS), an NMAC could occur. We see from the control structure that the ground controller has no information about what advisory TCAS has given to the pilot. The system designers must resolve this potential conflict in some way, perhaps requiring the pilot to always follow the TCAS advisory. Adding that requirement, however, also adds to the list of inadequate pilot control actions that must be examined, i.e., the pilot does not follow the TCAS advisory when a conflict occurs, that is, the pilot does not follow the normative procedures provided by the system designers and assumed in the analysis.

Inadequate Execution of Control Actions: A second way for constraints to be violated in the controlled process is if there is a failure or inadequacy in the transmission of control commands or in their execution (actuator fault or failure). A common flaw in the development safety control structure is that the safety information gathered or created by the system safety engineers (the hazards and the necessary design constraints to control them) is inadequately communicated to the system designers and testers.

The actuator for TCAS RAs (and ground controller advisories) is the pilot and the actuators for the pilot commands are the aircraft control systems. Clearly the aircraft control systems may fail, but a more subtle reason for the RA not to be executed is that it exceeds the aircraft performance limits. The performance limits of TCAS's own aircraft as well as the other aircraft must therefore be known to TCAS in order to provide executable resolution advisories.

Inadequate or Missing Feedback: The third flaw leading to system hazards involves inadequate feedback. No control system will perform better than its measuring channel. Important questions therefore arise about whether the controllers or decision makers (either automated or human) have the necessary information about the actual state of the controlled process to satisfy their objectives. This information is contained in their process models and updating these models correctly is crucial to avoiding accidents (1.2.2). Feedback may be missing or inadequate because such feedback is not included in the system design (3.1), flaws exist in the communication channel (3.2), the feedback is not timely (3.3), or the measuring instrument operates inadequately (3.4).

As an example, the sensors that provide critical information to TCAS about the other aircraft's state may not work correctly. Alternatively, they may be working, but the information received may be incorrect, either being garbled enroute or being incorrectly sent by the other aircraft. In addition, the transmission may be lost and no information may be received. Each of these cases will need to be accounted for in the design in some way.

SpecTRM-RL Completeness Criteria: Many of the completeness criteria previously identified by Jaffe and Leveson [1991] and incorporated into the SpecTRM-RL language design apply to the accident factors identified for an STPA (see Figure 8). Therefore, the SpecTRM-RL language already includes most of the facilities to model the critical factors, for example, feedback information. This fact is not surprising because the completeness criteria were originally derived, at least partially, from common causal factors in system accidents and basic system engineering principles. STAMP comes from systems theory, which is the foundation for system engineering principles.

Most of the completeness criteria use information that is included in a SpecTRM-RL specification and can be directly used in the hazard analysis, such as the properties of feedback loops and control command reversals. The completeness criteria that are not built into the SpecTRM-RL language were defined in terms of the formal state-machine model that underlies the language (the RSM), which should simplify the construction of tools to detect these forms of incompleteness. For example, the loss of a critical input sensor is a factor considered in STPA analysis. In the completeness criteria, we defined path robustness in terms of soft and hard failure modes [Jaffe et.al. 1991, Leveson 1995]. A *soft failure mode* is one in which the loss of the ability to receive a particular input I *could* inhibit the software from providing an output with a particular value while a *hard failure mode* involves the loss of the ability to receive an input I that *will* prevent the software from producing that output value. One of our completeness criteria related to safety is that: *Soft and hard failure modes should be eliminated for all hazard-reducing outputs. Hazard-increasing outputs should have both soft and hard failure modes*.

Soft and hard failure modes can be identified from the SpecTRM-RL model itself; the process does not require searching the entire reachable state space. Properties analyzable directly on the metalanguage for describing the state space (such as SpecTRM-RL) will usually involve simpler tools than those involved in searching reachability graphs. The design of the specification language obviously can affect how easy to use and efficient these analysis tools can be and how easily and efficiently the modeling language can be used in model-based hazard analysis.

Chapter 4: Comparison of STPA to Traditional Hazard Analysis Approaches

In this section we compare STPA to the most popular current approaches to hazard analysis and evaluate the results of STPA by comparing them to a high-quality fault tree analysis of the same system.

Comparison: Computational organizational models of risk management that apply to complex systems and throughout the life cycle of design, manufacturing, operations, and maintenance must be able to handle system accidents, organizational factors in accidents, software, human error and complex decision making, and adaptation. The standard techniques used in safety engineering, such as fault tree analysis, event tree analysis, HAZOP, and various types of failure analysis (e.g., Failure Modes and Effects Criticality Analysis), do not effectively handle these factors in accidents. STPA was designed to include such factors in the analysis. It can do this because while almost all current hazard analysis techniques are based on an event-chain model of accidents, STPA instead uses a model of accidents (STAMP) that is based on systems theory. Event chain models consider only direct relationships between events, whereas systems theory allows considering more complex feedback relationships and hierarchy theory. In fact, systems theory was developed in order to allow engineers to deal with complex systems and provides the theoretical foundation for system engineering.

Some widely used hazard analysis techniques, like FMECA (Failure Modes and Effects Criticality Analysis) were derived from bottom-up component reliability models of accidents and are not effective in dealing with system accidents. FMECA is used widely in NASA (but often incorrectly labeled as FMEA). For example, FMECA was used to identify the approximately 5000 criticality 1 and 1R[2] items in the Space Shuttle. Because these bottom-up techniques concentrate on component failure, they miss important risks associated with individual components operating without failure (according to specification) but interacting dysfunctionally with other components and interactions among multiple failures. Top-down techniques are needed to handle dysfunctional interactions and system accidents. They also are needed to handle the types of human "error" that does not involve simple slips.

Unlike the other analysis methods, MORT does incorporate management factors into the analysis. While MORT has primarily been used in accident investigation, it could potentially be used for hazard analysis. The checklist questions are so general, however, it is not clear how useful it would be. STPA starts from the specific system hazards and assigned responsibilities (see Chapter 5) and seems like it would be more useful.

Most hazard analyses (with the exception of HAZOP) are performed on system models contained only in the minds of the analysts. For complex systems, these mental representations become difficult to create and to use for analysis. They are also difficult to maintain as changes

[2] Criticality 1 items are defined as those items whose failure can cause the loss of the Shuttle and its crew. Criticality 1R items have redundancy and thus their failure alone cannot cause the loss. Criticality 2, 2R, 3, 3R etc. refer to components whose failure can only lead to less catastrophic losses.

are made, and they create problems in communication due to the lack of any concrete basis for ensuring that a common model is being used. In contrast, STPA and HAZOP use concrete models of the process. The HAZOP process has some similarities with STPA, but HAZOP employs physical piping and wiring diagrams (the physical structure) whereas STPA uses functional control models.

Most important, HAZOP is based on a model of accidents that views the causes as deviations in system variables whereas STPA uses a more general model of inadequate control. The deviation-based keywords used in HAZOP *(more of, less than*, etc.) are appropriate for certain types of hazards in the chemical process industry (where the technique originated), but not appropriate for other types of systems. These limitations of HAZOP make it difficult to extend to complex systems including software, etc. All attempts to do so have been unsuccessful and have simply reduced to a FMEA on the software components of the system.

In essence, HAZOP is an event-based model where the events of concern are deviations of process variables. STPA, in contrast, defines accident causation as a lack of enforcement of safety-related constraints on system behavior. These constraints might involve limitations on deviations in process variables and equipment behavior, making the results of HAZOP a subset of STPA. But STPA also includes constraints on the potential *interactions* among components (emergent properties of systems) and focuses on the *control* over the process variables rather than simply the effects of the lack of control, i.e., the deviations themselves. Therefore, STPA better handles the control functions in a system, such as software, operators, and management, and the accident factors (see Figure 8) used in STPA apply to more general types of systems than the HAZOP checklist. Another way of thinking about this is that HAZOP is still event-based, but limits the events to deviations in process variables whereas STPA (like STAMP) focuses on the control over these events or deviations and how that control could be flawed and lead to accidents.

A final difference is that while HAZOP uses models, they are not formal (analyzable) or executable. Because SpecTRM-RL is a formal modeling language, the potential exists for providing automated tools to assist the analyst through simulation (execution of the models) and various types of formal analysis. Simulation of the SpecTRM-RL models provides a means for examining the relationship between organizational decisions and system design (hardware, software, and operational procedures) that can be used in the analysis of system risk and resiliency. Although there are other modeling languages for hardware, software, human operator procedures, and organizational behavior, we know of no other modeling language that includes all of these factors and thus allows modeling the entire socio-technical system. Because the SpecTRM-RL language is designed to allow easy integration with other tools (through an API), the factors that it does not model (such as human fatigue, stress, workload, etc.) can be provided through integration with other modeling tools. The result would be a comprehensive system modeling of a type that is far beyond the current state-of-the-art.

Evaluation: Few evaluations of hazard analysis techniques have ever been done. Given their widespread use, the small amount of careful evaluation is surprising, but the difficulty of doing such evaluations is an important factor. One way to evaluate techniques is to compare the results

of the analysis with the actual events occurring during commissioning and operations. A second approach that has been used is to collect incident and accident information in real systems and evaluate which factors are found (or potentially could be found) by different analysis techniques. A third approach that has been used is to perform different types of analyses on the same system and compare the causal factors identified.

Because we had no access to the information required for the first two approaches, we chose the third approach and selected fault trees as the technique to evaluate our STAMP-based hazard analysis against. Fault trees are the most powerful of the event-chain analysis methods and also the most widely used. In order to prevent bias in the way the hazard analysis was performed on the system, we selected a fault tree that had been created by experts for a complex system, i.e., the MITRE fault tree for TCAS II created under contract to the FAA. This fault tree is the best we have ever seen for a complex software-intensive system—it is significantly better than other standard industrial fault trees and therefore provided a high-quality yardstick. We hypothesized that the STAMP-based hazard analysis would encompass everything covered by the MITRE TCAS II fault tree and hopefully more.

The MITRE TCAS fault tree examined only one hazard, a near midair collision (NMAC), but for that hazard alone has nearly 300 boxes. The fault tree considers TCAS, the pilot, and the ground controller. One thing to note is that despite the large number of factors considered, including both pilot and controller errors, the factors involved in the worst TCAS-related accident (the midair collision over Lake Constance) did not appear in the MITRE fault tree. In the accident, one of the pilots received different advisories from TCAS and the ground-based air traffic controller. The pilot selected the ground controller's incorrect maneuver over that of TCAS. This problem is what STAMP and STPA labels as a coordination problem, and it is included in an STPA hazard analysis of TCAS II. It is not fair to make too much of this difference between STPA and FTA due to the fact that we had the advantage of knowing about the accident and hindsight bias might be involved. However, it is also true that STPA includes this type of coordination problem specifically in the analysis process whereas it is somewhat awkward and difficult to include it in a fault tree because of the nature of the linear chains of events that fault trees use to specify accident scenarios.

An important part of the explanation for the Lake Constance accident is that the ground controller has no direct information about the TCAS advisories provided to the pilots (in STAMP terminology, the ground controller process model does not include this controlled process variable). The designers of TCAS knew that this omission might lead to problems, but they made the decision not to provide the ground controller with the aircraft's TCAS resolution advisory for technical reasons. STPA would have allowed them to investigate the safety of this design decision through simulation and analysis.

Unlike most industrial uses of fault tree analysis, the MITRE team used the same starting point as used in STPA, i.e. identifying the incorrect control actions of each system component. Therefore the TCAS FTA is similar to the TCAS STPA except that the FTA is less comprehensive. That is, STPA included more inadequate control actions and provided much more complete information about why that action might occur. The FTA stopped investigating the cause of a potential inadequate behavior before STPA did. Sometimes the fault tree branch

was ended with the explanation: "Not developed further because perceived to be a human factors-dependent fault." STPA considers such human factors issues. For example, the MITRE fault tree stops with "Pilot takes an action that does not avoid the NMAC because of information provided by the cockpit traffic advisory display" and labels this fault as a human-factors problem. STPA would continue with this scenario by examining what types of information might be missing or incorrect or misleading and why the information might be missing, incorrect, or misleading.

The less complete analysis did not just concern human factors. As another example of the FTA stopping before STPA did, a leaf node in the MITRE fault tree is labeled "TCAS does not issue an *Advisory Not OK*". First note that this feature is no longer in TCAS (and was not at the time the system was certified); reversals were substituted for Advisory Not OK messages several years before certification. The integration of the SpecTRM-RL model and the hazard analysis into the system specification in the SpecTRM toolset (which uses intent specifications [Leveson 2000]) provides the traceability to detect when changes have been made and analyses must be updated.

More important is the lack of investigation of why the reversal (or *Advisory Not OK* message) might not occur. Reversals are critical and were added to TCAS when a serious safety problem was considered surprisingly late in the system development. A reversal is issued when the other aircraft does not follow its TCAS resolution advisory and it is necessary to change TCAS's own aircraft advisory to avoid a collision. The seriousness of this problem is evidenced by the fact that the Lake Constance collision resulted from exactly this scenario, although the press accounts never mentioned the reversal possibility or why the DHL plane did not reverse (e.g., did TCAS not issue such a reversal or did the DHL pilot not follow it?) when the Tupelov pilot did not follow his TCAS advisory. The STPA hazard analysis investigated this problem in more depth than the MITRE fault tree.

We believe the differences in completeness arose because STPA has a set of criteria to follow to identify the problems whereas FTA is dependent on the imagination of the analysts. In fact, much of the STPA analysis could be displayed in a fault tree format and the STPA process could be used to generate fault trees. However, because almost none of the boxes that are generated for software intensive systems of this type are quantifiable (although the MITRE reports claims to have quantified the fault tree), there does not seem to be any reason to format the analysis results as a tree. We believe much more readable and usable formats are possible, perhaps based on standard notations used in control theory. A goal of our future research is to design such a format for recording and presenting the results of STPA. In addition, the fault tree format would not be suitable for some of the more complex, non-linear relationships between fault conditions included in STPA.

Chapter 5: Applying STPA to the Entire Socio-Technical System

Chapter 3 described and illustrated the use of a STAMP-based hazard analysis technique, STPA, on the technical and human operator parts of the system. Chapter 4 compared it to standard hazard analysis techniques and demonstrated its superior completeness versus fault tree analysis, the most commonly used hazard analysis technique. But it still remains to be shown that STPA can be applied to the entire socio-technical system, including the managerial and regulatory aspects. In this chapter we illustrate this capability using a Canadian public water system in the province of Ontario. This particular system was chosen because it contains computers, hardware, human operators, management decision-making, and government regulatory components. We also knew a lot about the safety control structure from a very comprehensive accident report that provided information about safety of public water systems. Using a real example makes the results more realistic than would result from a totally fabricated example.

In contrast to the TCAS analysis in Chapter 3, we performed the hazard analysis from scratch and designed an effective safety control structure rather than analyzing the risk inherent in an existing design. At the end, we compare our designed safety control structure with the actual one that existed at the time of a serious water contamination accident.

SpecTRM-RL, like most models, describes the static structure of systems. In order to handle the organizational components of open systems and the adaptation and changes that occur over time for such human-centered systems, we needed to be able to model dynamic system aspects. For this we used system dynamics models [Sterman 2000].

System Dynamics Models: By focusing on the events immediately preceding accidents, event chains treat a system as a static, unchanging structure. But systems and organizations continually experience change and adaptation to existing conditions. Systems dynamics models are one way to describe dynamic change in systems. They have been used to examine the potential undesired consequences of organizational decision-making.

As noted in the description of the STAMP model of accidents, a system's defenses or safety controls may degrade over time due to changes in the behavior of the components of the safety control loop. The reasons for the migration of the system toward a state of higher risk will be system specific and can be quite complex. In contrast to the usually simple and direct relationships represented in event-chain accident models, most accidents in complex socio-technical systems involve relationships between events and human actions that are highly non-linear and contain multiple feedback loops. The prevention of accidents in these systems therefore requires an understanding not only of the static structure of the system (the *structural complexity*) and of the changes to this structure over time (the *structural dynamics*), but also the dynamics behind these changes (the *behavioral dynamics*).

SpecTRM-RL models capture the static control structure and are useful in performing a hazard analysis that examines complex control structures and the dynamics or structural changes (failures and dysfunctional interactions) that occur over time in these structures, but not the behavioral dynamics, i.e., the dynamic processes behind these structural changes. We are

adapting system dynamics techniques to model and understand the behavioral dynamics, i.e., the dynamic processes behind the changes to the static safety control structure: how and why the safety control structure might change over time, potentially leading to ineffective controls and unsafe or hazardous states.

The field of system dynamics, created at MIT in the 1950's by Jay Forrester, is designed to help decision makers learn about the structure and dynamics of complex systems, to design high leverage policies for sustained improvement, and to catalyze successful implementation and change. System dynamics provides a framework for dealing with dynamic complexity, where cause and effect are not obviously related. It is grounded in the theory of non-linear dynamics and feedback control, but also draws on cognitive and social psychology, organization theory, economics, and other social sciences [Sterman 2000]. System dynamics models, like SpecTRM-RL are formal and can be executed. The models and simulators help to capture complex dynamics and to create an environment for organizational learning and policy design. The combination of STPA, SpecTRM-RL and systems dynamics models could provide an extremely powerful integrated risk management approach that goes far beyond what is possible using current techniques.

System dynamics is particularly relevant when analyzing system accidents. The world is dynamic, evolving, and interconnected, but we tend to make decisions using mental models that are static, narrow, and reductionist. Thus decisions that might appear to have no effect on safety—or even appear to be beneficial—may in fact degrade safety and increase risk. System dynamics makes it possible, for example, to understand and predict instances of policy resistance or the tendency for well-intentioned interventions to be defeated by the response of the system to the intervention itself. In related but separate research, Marais and Leveson are working on defining archetypical system dynamic models often associated with accidents [Marais and Leveson, 2003].

System behavior in system dynamics is modeled by using feedback (causal) loops, stock and flows (levels and rates), and the non-linearities created by interactions between system components. In this view of the world, behavior over time (the dynamics of the system) can be explained by the interaction of positive and negative feedback loops [Senge 1990]. The models are constructed from three basic building blocks: positive feedback or reinforcing loops, negative feedback or balancing loops, and delays. Positive loops (called reinforcing loops) are self-reinforcing while negative loops tend to counteract change. Delays introduce potential instability into the system.

Figure 10a shows a *reinforcing loop*, which is a structure that feeds on itself to produce growth or decline. Reinforcing loops correspond to positive feedback loops in control theory. An increase in variable 1 leads to an increase in variable 2 (as indicated by the "+" sign), which leads to an increase in variable 1, and so on. The "+" does not mean that the values necessarily increase, only that variable 1 and variable 2 will change in the same direction. If variable 1 decreases, then variable 2 will decrease. A "–" indicates that the values change in opposite directions. In the absence of external influences, both variable 1 and variable 2 will clearly grow or decline exponentially. Reinforcing loops generate growth, amplify deviations, and reinforce change [Sterman 2000].

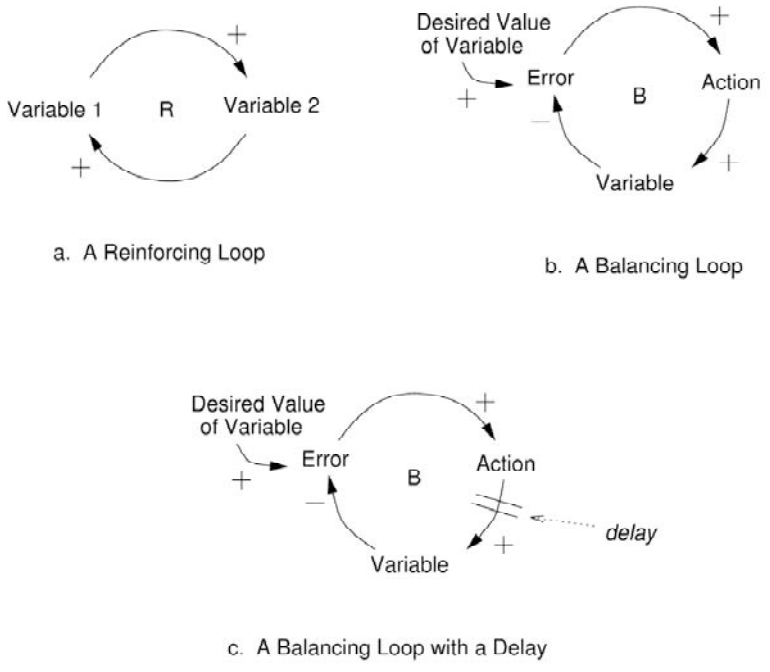


Figure 10 – The Three Basic Components of System Dynamics Models

A *balancing loop* (Figure 10b) is a structure that changes the current value of a system variable or a desired or reference variable through some action. It corresponds to a negative feedback loop in control theory. The difference between the current value and the desired value is perceived as an error. An action proportional to the error is taken to decrease the error so that, over time, the current value approaches the desired value.

The third basic element is a delay, which is used to model the time that elapses between cause and effect. A delay is indicated by a double line as shown in Figure 10c. Delays make it difficult to link cause and effect (dynamic complexity) and may result in unstable system behavior. For example, in steering a ship there is a delay between a change in the rudder position and a corresponding course change, often leading to over-correction and instability.

The STPA of socio-technical systems starts with a static SpecTRM-RL model of the safety control structure and then uses system dynamics modeling to predict and explain changes in that structure over time.

A Hazard Analysis of the Ontario (Canada) Public Water System: The system hazard to be examined is public exposure to health-related contaminants, such as E. coli, Campylobacter, and Cryptosporidium through the public drinking water system. The goal of the public water safety structure then is to prevent the exposure of the public to water with levels of contaminants that can affect health.

Public Water System Safety Constraints:

1. Water quality must not be compromised at levels that can negatively impact public health.
2. Public health measures must reduce the risk of exposure if water quality is compromised (for example, the public must be informed about the need to boil water coming from the public water supply).

The basic organizational structure of the public water system is shown in Figure 11. The physical processes being controlled are public health, the public water system, and the wells from which the water is obtained. The hazard analysis will show that controls over the environment, i.e., the water resources, are also necessary to reduce risk to an acceptable level. Public health is controlled by the Ministry of Health through the local Department of Health. The water system and the wells themselves are controlled by the local Public Utilities Commission (PUC) through an operations manager and human operators. The PUC is controlled in turn by the PUC commissioners. Government oversight and control of local water systems is the responsibility of the Ministry of the Environment. Control over water resources spans the Ministry of the Environment and the Ministry of Agriculture, Food, and Rural Affairs. The three ministries are controlled by the Provincial Government through budgets, laws, and the political appointments of the heads of each ministry.

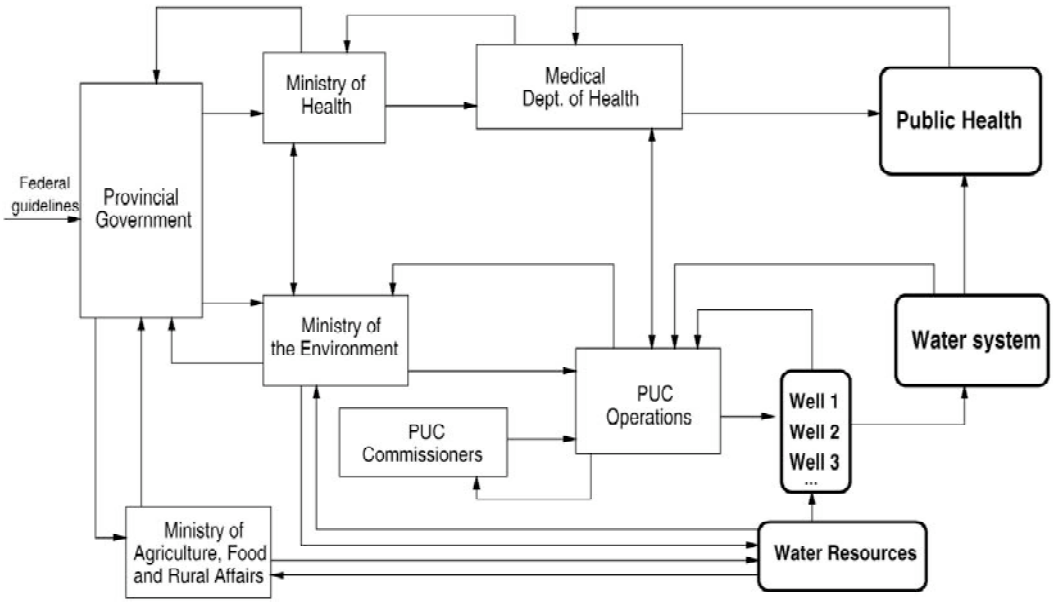


Figure 11 – The Public Water Safety Control Structure in Ontario. Lines going into the left of a box are control lines. Lines from or to the top or bottom of a box represent information, feedback, or a physical flow. Rectangles with sharp corners are controllers while rectangles with rounded corners represent physical plants.

Figure 11 shows the operations control structure (see Figure 4). The development control structure is not shown but would eliminate some of these boxes (e.g., those related to the Health Ministry) and would add design engineers and a construction company. More components may

be added to the operations control structure (than are shown in Figure 11) as the hazard analysis progresses and their need is determined.

The analysis starts with the physical process being controlled. Wells are used to obtain drinking water. Contaminants may infiltrate the water sources for the wells, and these contaminants can be transported into a well if the location of the well does not provide an adequate natural barrier or filtration in the overburden between the ground surface and the aquifer.

In general, there are four ways to manage the risk associated with a hazard:

1. Eliminate the hazard from the system design,
2. Reduce its likelihood of occurring,
3. Control it once it occurs, and
4. Minimize damage if it cannot be controlled.

System safety engineering tries first to eliminate and if that is not possible to mitigate the hazard by reducing the likelihood of it occurring and by controlling it if it does. There must also be contingency plans provided in case the attempts to control the hazard are ineffective. In this example, an obvious constraint on the water system design engineering process is that it must eliminate or reduce the potential for contamination. Elimination is clearly the preferred strategy, but is not always possible due to physical limitations or the need for unacceptable tradeoffs with other goals. For our water system example, placing the wells in locations where contamination cannot occur can potentially eliminate the threat of contamination from ground surface sources, but this approach may not be feasible. The likelihood of the hazard can be reduced through construction techniques, the installation of chlorination and other purification facilities, and land use policies enforced on the area potentially affecting the aquifer. If contamination does occur, it can be controlled through monitoring and sampling (to detect the contamination) and remedial measures (e.g., super-chlorinating the water to reduce the contamination). Finally, if the contamination does get into the water supply, the public must be warned (e.g., issuing a boil water advisory).

A risk-driven design process would consider these options starting from the earliest design phases. The choices will have important implications for the physical design and operation of the municipal water systems. Consider the physical design first. It must eliminate or reduce the potential for contamination. If the potential cannot be eliminated, then the design must allow for practical surveillance procedures to detect contamination (design for operability) and for chlorination. To exercise adequate control over public water safety, the design features chosen to mitigate the hazard should require approval by the regulatory agency (in this case, the Ministry of the Environment) and/or the imposition of design standards. For example, the regulatory authorities may require the installation of automated monitors of chlorine residuals and turbidity.[3] Special oversight of the construction process and testing of the safety-critical components should be the responsibility of either the Ministry of the Environment or the local PUC or both, perhaps through the hiring by the PUC of expert consultants on water safety. Information accumulated about the safety-related features of the design during development

[3] A decrease in the chlorine residuals is an indication that contaminants are overwhelming the chlorine levels and contingency measures are required.

must be archived by the regulators and operators and used during operations for monitoring, maintenance, and modification activities.

Operability is an important consideration in the physical design. During the design process, the safety constraints related to the identified system hazards and safety constraints that must be maintained during PUC operations are identified. For the first system safety constraint above, the corresponding local operations (PUC) constraint is that a level of chlorine must be maintained such that any contaminants in the raw water are inactivated. In addition, the chlorine residual after 15 minutes of contact time must be no more than 0.5 mg/L of water.

Responsibility for maintaining the second constraint, i.e., reducing the risk of exposure to water that has been contaminated, is assigned in this design to the Ministry of Health and its local Medical Departments of Health. This responsibility for identifying the contamination and initiating public safety measures could be assigned elsewhere—the allocation of functionality and responsibility to system components is an important part of early system design decisions and is no different for organizational components than for physical components. In each case, risk-driven design uses the information obtained from the hazard identification and hazard analysis process to provide input into these early design decisions so that risk can be minimized. If the use of safety-related information does not start early in the design process, then the choices for eliminating or minimizing risk in the design will most likely be reduced and a higher risk may have to be accepted than would have been necessary if the safety-related information had been considered in the early design choices and tradeoffs.

Once the assignments of responsibility for each of the system safety constraints are made to each of the system components, the next steps are (1) to identify required control actions for each of the system control components, (2) to build SpecTRM-RL models for these components, and then (3) to perform STPA using the models, as described in Chapter 3. The hazard analysis being performed in parallel with the system design will provide inputs to the design decision and tradeoff process. The on-going hazard analysis may result in safety-critical control actions being added to the emerging design and additional control components may be added. Note that the exact same process is used whether the components are technical or managerial.

To use SpecTRM-RL in this process, the control action (outputs) are identified first and then the conditions for issuing the action are defined using AND/OR tables and the other SpecTRM-RL features. The information required to perform these control actions will be identified in the definition of the control logic, and this information is then added to the emerging SpecTRM-RL *plant* (process) model. The process model in turn defines the type of feedback information that is required to make the safety-related decisions and thus assists in the design of feedback loops. For example, the water system operators are responsible for checking the level of residuals and turbidity in the chlorinated water. To carry out this task, the operators will need to obtain water samples, and those samples must be tested. Only large municipalities will be able to afford to maintain their own testing laboratories, so another component will need to be added to the control structure, i.e., water testing laboratories. The MOE will also need access to these laboratory results in order to execute their responsibility for oversight of water quality and to make sure the water quality standards are being adequately maintained and are not degrading.

Therefore, feedback loops from the water-testing laboratory to the MOE (and perhaps other places) must also be added to the design.

Two different designs are possible: (1) the testing lab may report to the local PUC, which then reports the results to the MOE or (2) the testing lab may report the results to both the local PUC and the MOE and perhaps to other control components, such as the MDH, as determined necessary in the STPA of the other system components. Here is where STPA can be helpful in making decisions. The control loop that implements the feedback about the local municipality's water testing results is critical. STPA is used to identify how flaws in the parts of the control loop could lead to ineffective control action being taken by the MOE. As described in Chapter 3, the use of SpecTRM-RL models for walkthroughs, simulation, and formal analysis assist in performing the hazard analysis. Design features might be added to mitigate any potential flaws found by STPA, and alternative designs can be evaluated for their comparative robustness and resiliency. For example, having the water testing labs provide their results directly to the MOE rather than through the local PUC eliminates the potential for incorrect reporting by the PUC (this could happen for a variety of reasons, including deliberate falsification of the reports due to various pressures on the PUC). Another decision is whether to establish government water testing laboratories or to use private labs. Clearly, other factors other than simply risk are involved in most of these decisions, such as political and budgetary considerations. In our example, either private or government labs could be used to test the water, but in either design the feedback to the MOE (and other parts of the control structure) must be ensured in the design of the processes and feedback loops.

Again, it is important to note that although organizational design is being stressed in this example, the same applies to the technical and automated components of the system. While the hazard analysis of TCAS II described in Chapter 3 was performed on the existing design of the system, clearly a risk-driven design process could have been applied during TCAS II development with clear advantages in terms of optimizing the design and finding weak points when they could be changed. After-the-fact hazard analysis limits the possibilities for controlling risk.

We now have the basic physical design and both development and operations control structures for the first of the system safety constraints, i.e., water quality must not be compromised at levels that can negatively impact public health. The second constraint must also be enforced, i.e., public health measures must reduce the risk of exposure if water quality is compromised. Responsibility for this constraint lies with the other branch of the control structure, namely, the Ministry of Health and its local Medical Department of Health (MDH). The local MDH must determine when contamination has occurred at a level that requires public notification, ensure that the public is made aware of the hazard, and ensure they have the appropriate knowledge about how to protect themselves, e.g., to boil all drinking water from the public water system before using it. For this, the MDH has various defined control actions.

Clearly, the state of the water system (in terms of contamination) is a critical part of the process model of the responsible individuals in the MDH. How is the value of that state variable (to use SpecTRM-RL terminology) determined? Information could come from the water testing laboratory, the managers of the PUC, hospitals, local medical doctors, and complaints by the

public. Feedback loops need to be established for these various sources of information. Issuing boil water advisories that are false alarms can have negative long-term consequences, so multiple sources of information should be used and an algorithm determined for how the information will be used to make a decision, particularly when the input is conflicting or unclear. The means for disseminating the health advisories must also be included. STPA can be used to identify ways in which the parts of the control loop could delay or corrupt the feedback information or hinder the dissemination of the boil water advisories. STPA also requires that the control algorithms themselves be analyzed for flaws. The Ministry of Health is responsible for making sure the local offices have adequate staffing, training, information, etc. to perform their responsibilities (provide adequate control actions). And so on.

As noted in Chapter 1 as one of the reasons for a new approach to hazard analysis, humans do not necessarily follow the specified procedures. Ensuring that an effective procedure exists is important as well as ensuring that the information required to make safe decisions and control actions is available to the operator and decision makers. But that is only the first part of STAMP-based hazard analysis. The analysis must also consider (probably using human factors experts) likely deviations from the specified procedures. The use of system dynamics models will help here, as described in the next section. STPA can also be used to identify the unsafe decisions and control actions. That information can be used in training, in performing monitoring, and in designing protection into the system design against such unsafe action (although there are drawbacks to trying to limit the authority and potential control actions of human operators as has been found in the attempts to build aircraft automation that overrules pilot commands). Finally, the system models built in SpecTRM-RL can assist in determining the safety-related side effects of decisions on other components of the system and the safety of the system as a whole. Thus it can be used to analyze the impact of specific control actions or of potential changes to the system before they are implemented.

The experienced engineer will at this point (or earlier) note that engineers use the type of thinking being described when designing any system. The difference simply is that risk is made a "first-class" part of the early design process. While sometimes this early emphasis occurs, often the early design analysis concentrates on what the system is required to do, rather than what it is required *not* to do. The former is particularly common for systems where the required functions (the basic mission) do not involve safety, i.e., the system itself is not being build to maintain safety, and safety is simply a constraint on the way the required functions can be achieved. Even when safety is an early consideration in conceptual design, the process used is primarily ad hoc and can therefore easily be flawed. For complex systems, doing all this analysis in one's head is difficult and error prone. The process described in this report can be implemented using tools that help the design engineer manage the complexity involved and assist with the analysis and system engineering process.

Figure 12 shows an overall control system structure that can be derived from the application of STPA while designing the public water system. Of course, different design decisions could be made and alternative designs result. The process described in this chapter, however, allows making risk-informed decisions when design alternatives and tradeoffs are being considered. The complete model would include for each control component a description of the safety constraints that must be maintained by the controller in the behavior of the system components it

controls, the information in the control component's process model (models of the current state of the controlled components) and how that information will be updated, and the control actions and the logic used to determine when the control action should be taken.

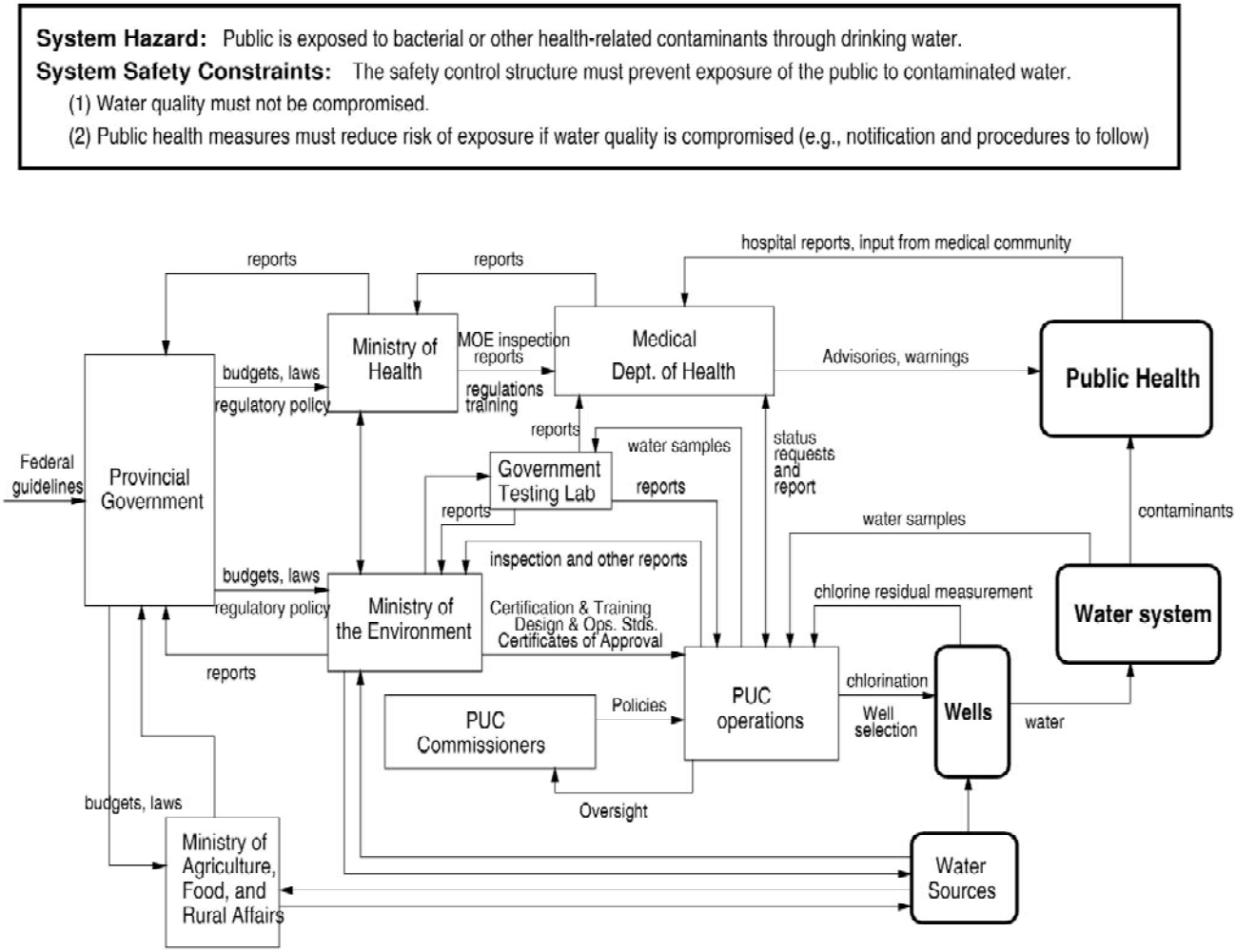


Figure 12 – An Ontario Water System Safety Control Structure
Resulting from STPA Analysis

We are not yet finished, however. Even if the system starts out with an effective safety control structure, adaptation, asynchronous evolution, and degradation can occur over time. The hazard analysis needs to identify unsafe changes and to design in operational controls to prevent or mitigate them and/or to design and implement performance measures to detect them. For that we use system dynamics models.

System Dynamics Model for Safety of the Ontario Public Water System: Figure 13 shows a system dynamics model for the Ontario public water system. The model focuses on the organizational factors and omits the physical process, although it could be included. Pressure to cut budgets and reduce government (exogenous variables external to the model) can lead to decreased oversight. The level of oversight affects training and certification as well as inspection and monitoring, both of which impact risk. The loop on the left says that as oversight decreases, the incident and accident rate will increase, which should decrease public pressure to reduce government regulation and red tape, thus leading to increased oversight and thence to decreases in the accident rate. The delay between changes in oversight and changes in the accident rate, however, introduces instabilities in the system (as is true in any control system). The lack of immediate feedback from incidents and accidents after oversight is reduced contributes to increased pressures to reduce oversight until the stage for a major tragedy is set.

Modeling the entire system dynamics is usually impractical. The challenge is to choose relevant subsystems and model them appropriately for the intended purpose. STAMP provides the guidance for determining what to model when the goal is risk management.

In complex systems, all dynamics, despite their complexity, arise from the two types of feedback loops described earlier in this chapter. In system dynamics terms, degradation over time of the safety control structure, as represented by reinforcing loops, would lead inevitably to an accident, but there are balancing loops, such as regulation and oversight that control those changes. One of the factors in our model is political pressure to reduce regulation and government red tape. As feedback and monitoring controls are reduced, the mental model of the central government leaders and the ministries about the current state of the water system can become increasingly divorced from reality. A belief that the water quality controls are in better shape than they actually are can lead to disregarding warnings and continued reduction in what is regarded as unnecessary regulation and red tape.

Accidents occur when the balancing loops do not adequately overcome the influences degrading the safety controls. Understanding how and why this degradation can occur (why risk may increase over time) is an important part of designing to prevent accidents, i.e., establishing effective safety control structures that include controls over evolution toward higher risk over time. These controls may simply involve monitoring and alerts to identify when the socio-technical system is moving toward a state of unacceptable risk or they may involve concrete checks and balances that prevent or inhibit potentially dangerous changes from occurring.

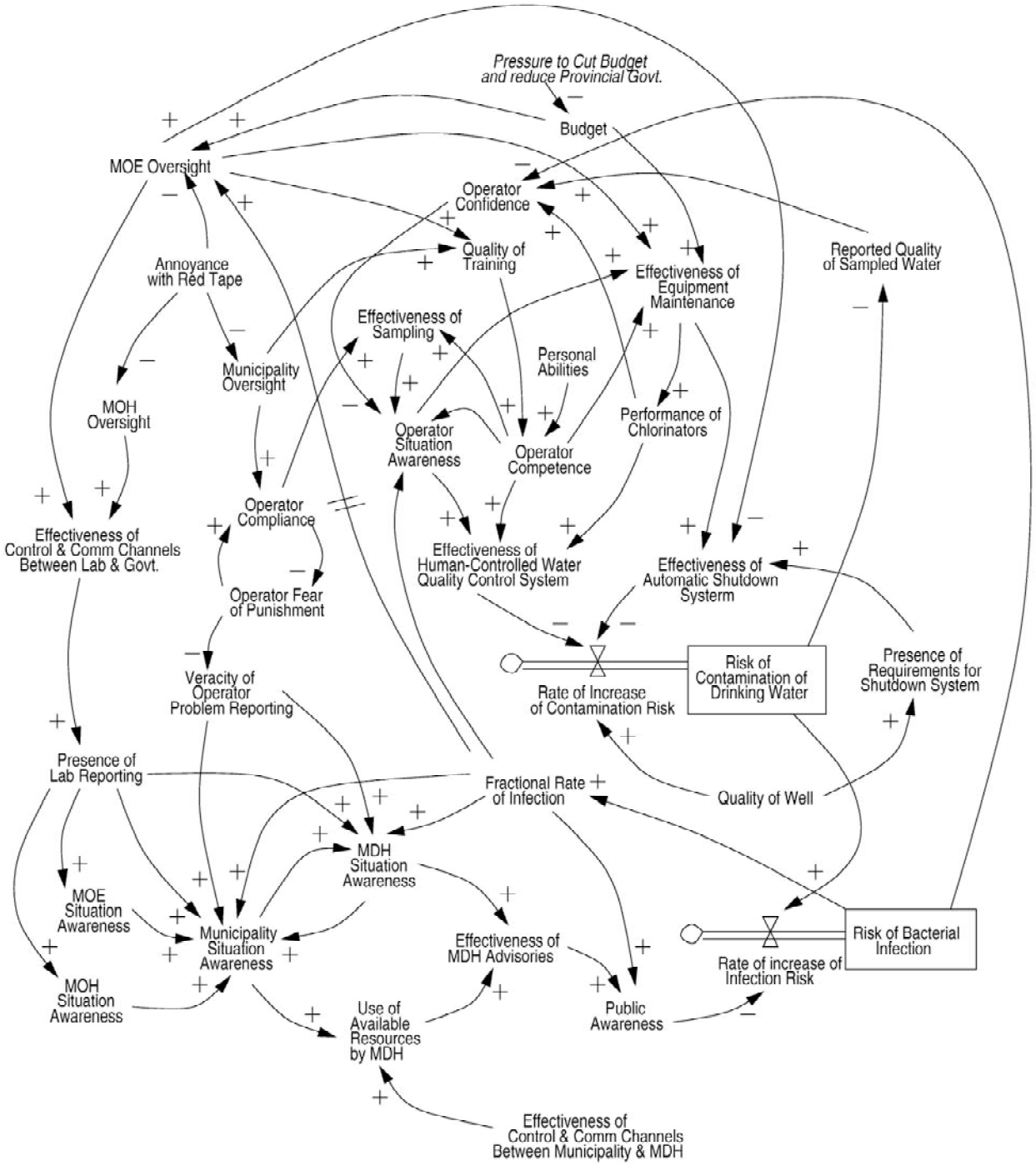


Figure 13 – A System Dynamics Model of Water System Safety
(Adapted from Leveson, Daouk, Dulac, and Marais, 2003)

Reinforcing and balancing loops can be analyzed separately from the rest of the system dynamics diagram. For example, the Public Awareness balancing loop (shown in Figure 14) provides insight about the feedback relationship between an increase in ``Risk of bacterial infection'' and an decrease in the ``Rate of increase of infection risk.'' An increase in the risk of infection would increase the fractional rate of infection. Individuals experiencing intestinal symptoms would suspect water contamination and the public awareness of the water problem would increase. Becoming aware of the problem would lead people to stop drinking the water altogether or to boiling it first. These prevention measures reduce the rate of increase of infection risk and consequently, the risk of bacterial infection.

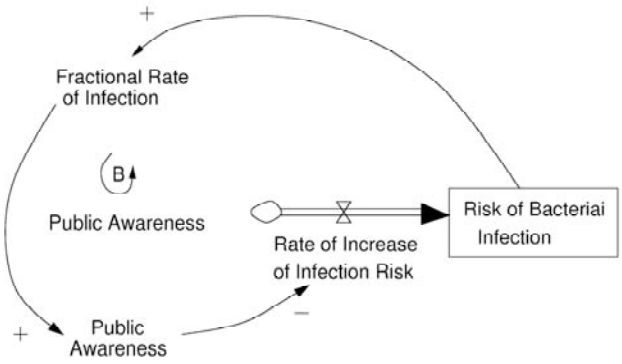


Figure 14 – The Public Awareness Balancing Loop

The public water system dynamics model in Figure 13 includes a number of exogenous variables (pressure to cut budgets, attempts by a conservative governments to reduce business and government red tape, etc.) that act as levers on the behaviors of the system. When these variables are changed without any consideration of the dynamics of the system being modeled, the effectiveness of the safety control structure can deteriorate progressively, with few if any visible signs. For instance, the attempts to reduce red tape can lead to a decrease in oversight by the ministries and municipalities. This decrease in oversight in turn can have a negative effect on the control and communication channels between the government and the laboratories performing water quality analyses. Lack of or disruptions in the communication of the results of water quality reports can lead to delays in the mobilization of resources needed to deal with contamination, diminishing effectiveness of the advisories issued, and thence an increase in risk of infection in the population.

Accident investigations often end with blame being assigned to particular individuals, often the system operators. System dynamics models show how the attitude and behavior of individuals is greatly affected by the rest of the system and how and why such behavior may change over time. For instance, operator competence depends on the quality of training, which increases with government oversight but may decrease over time without such oversight due to competing pressures. An operator's fear of punishment (which in the accident described in the next section led to the PUC operations manager lying about the existence of adverse water quality test reports) is balanced by compliance with existing rules and regulations. This compliance, in turn, is directly influenced by the extent of government oversight and by the government's response to similar behavior in the past.

Tools to simulate system dynamics models exist. They have limitations, however, that would be lessened by combining them with SpecTRM-RL models. The most serious of these is that the variables are all treated as continuous and simulations involve evaluating their changes using continuous equations. Combining systems dynamics models with SpecTRM-RL would allow much more comprehensive types of system modeling.

The chapter has so far described how STPA, SpecTRM-RL, and system dynamics models can be used to perform hazard analysis on the design of a public water system, accounting in the process for adaptation and degradation over time. The specific design decisions made, of course, are up to the designers and will involve considerations other than simply risk of violating safety goals and constraints. The next section shows how this process can go wrong.

A Public Water Accident in Ontario: It is instructive in evaluating both STAMP and its associated hazard analysis technique STPA to examine in detail an accident involving a real municipal water system in Ontario. In May 2000, some contaminants (largely E. coli and Campylobacter) entered the public water system of a small town in Ontario called Walkerton. As a result, 2300 people became ill (in a community of 4600 residents) and seven died.

The water safety control structure started out similar to the one designed above but with some potential weaknesses that were mitigated by the presence of various controls. As those controls weakened or disappeared over time, the entire socio-technical system moved to a state where a small change in the operation of the system or in the environment (in this case, unusually heavy rain) could lead to a tragedy. We describe the accident scenario in Walkerton, which had such serious consequences that an official government inquiry was launched. Other municipalities in Ontario, however, have also had water contamination incidents due to the inadequate risk management and controls in the current Ontario public water system safety control structure (Figure 15).

The Walkerton Public Utilities Commission (WPUC) operated the local water system. Stan Koebel was the WPUC's general manager and his brother Frank its foreman. In May 2000, the water system was supplied by three groundwater sources: Wells 5, 6, and 7. The water pumped from each well was treated with chlorine before entering the distribution system.

Walkerton Well 5 was built in 1978 and issued a Certificate of Approval by the MOE in 1979. At the time, the groundwater supplying the well was recognized as being vulnerable to surface contamination, but no explicit operating conditions were imposed by the regulatory authorities (*missing MOE control action*). Well 5 is a very shallow well: all of its water is drawn from an area between 5m and 8m below the surface. More significantly, the water is drawn from an area of bedrock, and the shallowness of the soil overburden above the bedrock along with the fractured and porous nature of the bedrock itself made it possible for surface bacteria to make its way to Well 5.

Although the original Certificate of Approval for Well 5 did not include any special operating conditions, over time MOE practices changed (*asynchronous evolution*). By 1992, the MOE had developed a set of model operating conditions for water treatment and monitoring that were

routinely attached to new Certificates of Approval for municipal water systems. There was no effort, however, to determine whether such conditions should be attached to existing certificates, such as the one for Well 5 (*missing MOE control action*).

The ODWO was amended in 1994 to require the continuous monitoring of chlorine residuals and turbidity for wells supplied by a groundwater source that was under the direct influence of surface water (as was Walkerton's Well 5). Automatic monitoring and shutoff valves would have mitigated the operational problems at Walkerton and prevented the deaths and illness associated with the E. coli contamination in May 2000 if the requirement had been enforced in existing wells. However, at the time, there was no program or policy to review existing wells to determine whether they met the requirements for continuous monitoring (*missing MOE control action*). In addition, MOE inspectors were not directed to notify well operators (like the Koebel brothers) of the new requirement nor to assess during inspections if a well required continuous monitoring (*MOE control flaw*). Stan and Frank Koebel lacked the training and expertise to identify the vulnerability of Well 5 themselves and to understand the resulting need for continuous chlorine residual and turbidity monitors.

Operating conditions should theoretically have been imposed by the municipality, the Walkerton Public Utilities Commissioners, and the manager of the WPUC. The municipality left the operation of the water system to the WPUC (*inadequate control actions*). The WPUC Commissioners, who were elected, became over the years more focused on the finances of the PUC than the operations (*asynchronous evolution*). They had little or no training or knowledge of water system operations or even water quality itself (*inadequate mental models*). Without such knowledge and with their focus on financial issues, they gave all responsibility for operations to the manager of the WPUC (Stan Koebel) and provided no other operational oversight.

The operators of the Walkerton water system did not intentionally put the public at risk. Stan Koebel and the other WPUC employees believed the untreated water was safe and often drank it themselves at the well sites (inadequate mental models). Local residents also pressed the WPUC to decrease the amount of chlorine used because they objected to the taste of chlorinated water (*hazardous inputs, inadequate control*).

Although Mr. Koebel knew how to operate the water system mechanically, he lacked knowledge about the health risks associated with a failure to properly operate the system and of the importance of following the MOE requirements for treatment and monitoring. This incorrect mental model was reinforced when over the years he received mixed messages from the MOE about the importance of several of its own requirements.

Before 1993, there were no mandatory certification requirements for water system operators or managers. Stan and Frank Koebel were not qualified to hold their positions within the WPUC, but they were certified in 1993 through a grandfathering scheme based solely on experience. They were not required to take a training course or to pass any examinations (*missing and inadequate control actions*).

After the introduction of mandatory certification in 1993, the MOE required 40 hours of training a year for each certified operator. Stan and Frank Koebel did not take the required amount of training, and the training they did take did not adequately address drinking water safety. The MOE did not focus the training on drinking water safety and did not enforce the training requirements (*missing control actions*).

The Koebel brothers and the Walkerton commissioners were not the only ones with inadequate training and knowledge of drinking water safety. Evidence at the Inquiry showed that several environmental officers in the MOE's local office were unaware that E. coli was potentially lethal and their mental models were also incorrect with respect to other matters essential to water safety.

Without regulations or oversight or enforcement of safe operating conditions, and with inadequate mental models of the safety requirements, operating practices have a tendency to change over time in order to optimize a variety of goals that conflict with safety. In the case of Walkerton, this change began almost immediately. The Inquiry report says that many improper operating practices had been going on for years before Stan Koebel became manager. He simply left them in place. These practices, some of which went back 20 years, included misstating the locations at which samples for microbiological testing were taken, operating wells without chlorination, making false entries in daily operating sheets, failing to measure chlorine residuals daily, failing to adequately chlorinate the water, and submitting false annual reports to the MOE (*inadequate "actuator" operation, incorrect feedback*).

All of these weaknesses in the control over the Walkerton (and other municipalities) water quality might have been mitigated if the source of contamination of the water had been controlled. A weakness in the basic water control structure was the lack of a government watershed and land use policy for agricultural activities that can impact drinking water sources. In fact, at a meeting of the Walkerton town council in November 1978 (when Well 5 was constructed), MOE representatives suggested land use controls for the area around Well 5, but the municipality did not have the legal means to enforce such land use regulations because the government of Ontario had not provided the legal basis for such controls.

Walkerton is at the heart of Ontario's Bruce County, a major farming area. Whereas the existing water quality infrastructure and physical well designs were able to handle the amount of manure produced when farms typically produced 50 or 60 animals at a time, the increase in factory farms (each of which might have 1200 hogs) led to runoff of agricultural contaminants and put pressure on the drinking water quality infrastructure (*asynchronous evolution*). At the time of the accident, the county had a population of only 60,000 people, but had 163,000 beef cattle and 100,000 hogs. A single 1200 hog factory farm can produce as much waste as 60,000 people and the entire animal population in the county at that time produced as much waste as 1.6 million people. This animal waste was spread on the fields adjacent to the farms, which could not absorb such massive quantities of manure. As a result, the groundwater and surrounding waterways were contaminated. At the same time, the spreading of manure had been granted a long-standing exemption from EPA requirements.

Annual reports of the Environment Commissioner of Ontario for the four years before the Walkerton accident included recommendations that the government create a groundwater strategy. A Health Canada study stated that the cattle counties of Southwestern Ontario, where Walkerton is located, are high-risk areas for E. coli infections. The report pointed out the direct link between cattle density and E. coli infection, and showed that 32 percent of the wells in rural Ontario showed fecal contamination. Dr. Murray McQuigge, the Medical Officer of Health for the BGOS Health Unit (and the man who handled the Walkerton E. coli outbreak) warned in a memo to local authorities that ``poor nutrient management on farms is leading to a degradation of the quality of ground water, streams, and lakes." Nothing was done in response to these warnings (*ignored feedback*).

The control structure quickly started to degrade even further in effectiveness with the election of a conservative provincial government in 1995. A bias against environmental regulation and red tape led to the elimination of many of the existing government controls over drinking water quality. A Red Tape Commission was established by the provincial government to minimize reporting and other requirements on government and private industry. At the same time, the government disbanded groups like the Advisory Committee on Environmental Standards (ACES), which reviewed ministry standards including those related to water quality. At the time of the Walkerton contamination, there was no opportunity for stakeholder or public review of the Ontario clean water controls (*feedback loops eliminated*).

Budget and staff reductions by the conservative government took a major toll on environmental programs and agencies (although budget reductions had started before the election of the new provincial government). The MOE budget was reduced by 42% and 900 of the 2400 staff responsible for monitoring, testing, inspection, and enforcement of environmental regulations were laid off. The official Walkerton Inquiry report concludes that the reductions were not based on an assessment of the requirements to carry out the MOE's statutory requirements nor on any risk assessment of the potential impact on the environment or, in particular, on water quality. After the reductions, the Provincial Ombudsman issued a report saying that cutbacks had been so damaging that the government was no longer capable of providing the services that it was mandated to provide. The report was ignored.

In 1996, the Water Sewage Services Improvement Act was passed, which shut down the government water testing laboratories, downloaded control of provincially owned water and sewage plants to the municipalities, eliminated funding for municipal water utilities, and ended the provincial Drinking Water Surveillance Program, under which the MOE had monitored drinking water across the province (*elimination of safety controls and feedback loops*).

The ODWO directed testing labs to report any indications of unsafe water quality to the MOE and to the local Medical Officer Of Health. The latter would then decide whether to issue a boil water advisory. When government labs conducted all of the routine drinking water tests for municipal water systems throughout the province, it was acceptable to keep the notification protocol in the form of a guideline under the ODWO rather than a legally enforceable law or regulation. However, the privatization of water testing and the exit of government labs from this duty in 1996 made the use of guidelines ineffective in ensuring necessary reporting would occur. At the time, private environmental labs were not regulated by the government. No criteria were

established to govern the quality of testing or the qualifications or experience of private lab personnel, and no provisions were made for licensing, inspection, or auditing of private labs by the government. In addition, the government did not implement any program to monitor the effect of privatization on the notification procedures followed whenever adverse test results were found (*inadequate control actions and missing feedback loop*).

At the time of privatization in 1996, the MOE sent a guidance document to those municipalities that requested it. The document strongly recommended that a municipality include in any contract with a private lab a clause specifying that the laboratory directly notify the MOE and the local Medical Officer of Health about adverse test results. There is no evidence that the Walkerton PUC either requested or received this document (*communication flaw*).

After laboratory testing services for municipalities were assumed by the private sector in 1996, the MOH Medical Department of Health for the Walkerton area sought assurances from the MOE's local office that the MDH would continue to be notified of all adverse water quality results relating to community water systems. It received that assurance, both in correspondence and at a meeting of representatives from the two agencies.

In 1997, the Minister of Health took the unusual step of writing to the Minister of the Environment requesting that legislation be amended to ensure that the proper authorities would be notified of adverse water test results. The Minister of the Environment declined to propose legislation, indicating that the ODWO dealt with the issue. On several occasions, officials in the MOH and the MOE expressed concerns about failures to report adverse test results to local Medical Officers of Health in accordance with the ODWO protocol. But the anti-regulatory culture and the existence of the Red Tape Commission discouraged any proposals to make notification legally binding on the operators of municipal water systems and private labs.

The testing laboratory used by Walkerton in May 2000, A&L Canada Laboratories East, was not aware of the notification guideline in the ODWO (*communication flaw*). In fact, they considered test results to be confidential and thus improper to send to anyone but the client, in this case, the WPUC manager Stan Koebel (*incorrect process model*). The MOE had no mechanism for informing private laboratories of the existing guidelines for reporting adverse results to the MOE (*missing control channel*).

Another important impact of the 1996 law was a reduction in the MOE water system inspection program. The cutbacks at the MOE negatively impacted the number of inspections, although the inspection program had other deficiencies as well.

The MOE inspected the Walkerton water system in 1991, 1995, and 1998. At the time of the inspections, problems existed relating to water safety. Inspectors identified some of them, but unfortunately two of the most significant problems—the vulnerability of Well 5 to surface contamination and the improper chlorination and monitoring practices of the PUC—were not detected (*inadequate "actuator" operation*). Information about the vulnerability of Well 5 was available in MOE files, but inspectors were not directed to look at relevant information about the security of water sources and the archived information was not easy to find (*inadequate control algorithm*). Information about the second problem, improper chlorination and monitoring

practices of the WPUC, was there to be seen in the operating records maintained by the WPUC. The Walkerton Inquiry report concludes that a proper examination of the daily operating sheets would have disclosed the problem. However, the inspectors were not instructed to carry out a thorough review of operating records (*inadequate control algorithm*).

The 1998 inspection report did show there had been problems with the water supply for years: detection of E. coli in treated water with increasing frequency, chlorine residuals in treated water at less than the required 0.5 mg/L, non-compliance with minimum bacteriological sampling requirements, and improper maintenance of training records.

The MOE outlined improvements that should be made, but desperately short of inspection staff and faced with small water systems across the province that were not meeting standards, it never scheduled a follow-up inspection to see if the improvements were in fact being carried out (*inadequate actuator operation, missing feedback loop*). The Inquiry report suggests that the use of guidelines rather than regulations had an impact here. The report states that had the Walkerton PUC been found to be in non-compliance with a legally enforceable regulation, as opposed to a guideline, it is more likely that the MOE would have taken stronger measures to ensure compliance—such as the use of further inspections, the issuance of a Director's Order (which would have required the WPUC to comply with the requirements for treatment and monitoring), or enforcement proceedings. The lack of any follow-up or enforcement efforts may have led the Koebel brothers to believe the recommendations were not very important, even to the MOE (*flawed mental model*).

The WPUC Commissioners received a copy of the 1998 inspection report but did nothing beyond asking for an explanation from Stan Koebel and accepting his word that he would correct the deficient practices (*inadequate control*). They never followed up to make sure he did (*missing feedback*).

The mayor of Walkerton and the municipality also received the report but they assumed the WPUC would take care of the problems. When the local Walkerton public health inspector read the report, he filed it, assuming that the MOE would ensure that the problems identified were properly addressed. Note the *coordination problems* here in an area of *overlapping control*, a major component of the accident factors used in STPA as defined in Chapter 3. Both the MOE and the local public health inspector should have followed up on the 1998 inspection report, but there was no written protocol instructing the public health inspector on how to respond to adverse water quality reports or inspection reports. The local Medical Director of Health assumed the MOE and the WPUC commissioners would do so. Again we see a classic example of flawed coordination where everyone assumes someone else is taking care of the problem. The MOE lacked protocols for follow-up, and the WPUC Commissioners by this time lacked expertise and relied on Stan Koebel to make the necessary changes. The Province's water safety control structure had clearly become ineffective.

A final important change in the safety control structure involved the drinking water surveillance program in which the MOE monitored drinking water across the province. In 1996, the Provincial government dropped E. coli testing from its Drinking Water Surveillance Program. The next year, the Drinking Water Surveillance Program was shut down entirely (*feedback loop*

eliminated). At the same time, the provincial government directed MOE staff not to enforce dozens of environmental laws and regulations still on the books (*control algorithm eliminated*). Farm operators, in particular, were to be treated with understanding if they were discovered to be in violation of livestock and waste-water regulations. By June 1998, the Walkerton town council was concerned enough about the situation to send a letter directly to the Premier (Mike Harris), appealing for the province to resume testing of municipal water. There was no reply.

MOE officials warned the government that closing the water-testing program would endanger public health. Their concerns were dismissed. In 1997, senior MOE officials drafted another memo that the government *did* heed. This memo warned that cutbacks had impaired the Ministry's ability to enforce environmental regulations to the point that the Ministry could be exposed to lawsuits for negligence if and when an environmental accident occurred. In response, the Provincial government called a meeting of the Ministry staff to discuss how to protect itself from liability, and it passed a Bill (The Environmental Approvals Improvement Act) which, among other things, prohibited legal action against the government by anyone adversely affected by the Environment Minister's failure to apply environmental regulations and guidelines.

Many other groups warned senior government officials, ministers, and the Cabinet of the danger of what it was doing, such as reducing inspections and not making the notification guidelines into regulations. The warnings were ignored. Environmental groups prepared briefs. The Provincial Auditor, in his annual reports, criticized the MOE for deficient monitoring of groundwater resources and for failing to audit small water plants across the province. The International Joint Commission expressed its concerns about Ontario's neglect of water quality issues, and the Environmental Commissioner of Ontario warned that the government was compromising environmental protection, pointing specifically to the testing of drinking water as an area of concern.

In January 2000 (three months before the Walkerton accident), staff at the MOE's Water Policy Branch submitted a report to the Provincial government warning that "Not monitoring drinking water quality is a serious concern for the Ministry in view of its mandate to protect public health." The report stated that a number of smaller municipalities were not up to the job of monitoring the quality of their drinking water. It further warned that because of the privatization of the testing labs, there was no longer a mechanism to ensure that the MOE and the local Medical Officer of Health were informed if problems were detected in local water systems. The Provincial government ignored this feedback.

The warnings were not limited to groups or individuals. Many adverse water quality reports had been received from Walkerton between 1995 and 1998. During the mid to late 1990s, there were clear indications that the water quality was deteriorating. In 1996, for example, hundreds of people in Collingswood (a town near Walkerton) became ill after cryptosporidium (a parasite linked to animal feces) contaminated the drinking water. Nobody died, but it should have acted as a warning that the water safety control structure had degraded. Between January and April of 2000 (the months just prior to the May E. coli outbreak), the lab that tested Walkerton's water repeatedly detected coliform bacteria—an indication that surface water was getting into the water supply. The lab notified the MOE on five separate occasions. The MOE in turn phoned the WPUC, was assured the problems were being fixed, and let it go at that (*inadequate control*

action). The MOE failed to inform the Medical Officer of Health, as by law it was required to do (*communication flaw*). One of the reasons for the delay in issuing a boil water advisory when the symptoms of E. coli contamination started to appear in Walkerton was that the latest report in the local Medical Department of Health's files of any problems with the water was over two years old (*incorrect mental model*). In May 2000, Walkerton changed its testing lab to A&L Canada who, as noted earlier, did not know about the reporting guidelines.

The Walkerton Inquiry report found that the decisions to remove the water safety controls in Ontario or to reduce their enforcement were taken without an assessment of the risks or the preparation of a risk management plan. The report says there was evidence that those at the most senior levels of government who were responsible for the decisions considered the risks to be manageable, but there was no evidence that the specific risks were properly assessed or addressed. We believe that the use of STPA, system dynamics models, and modeling and simulation tools like SpecTRM-RL and system dynamics simulation could have provided the ability to perform such a risk assessment. Of course, management that refuses to perform risk management activities will obviously sabotage any efforts to manage risk.

All of these changes in the Ontario water safety control structure over time led to the modified control structure shown in Figure 15. One thing to notice in comparing Figure 15 and Figure 12 is the omission of many of the feedback loops and controls in Figure 15.

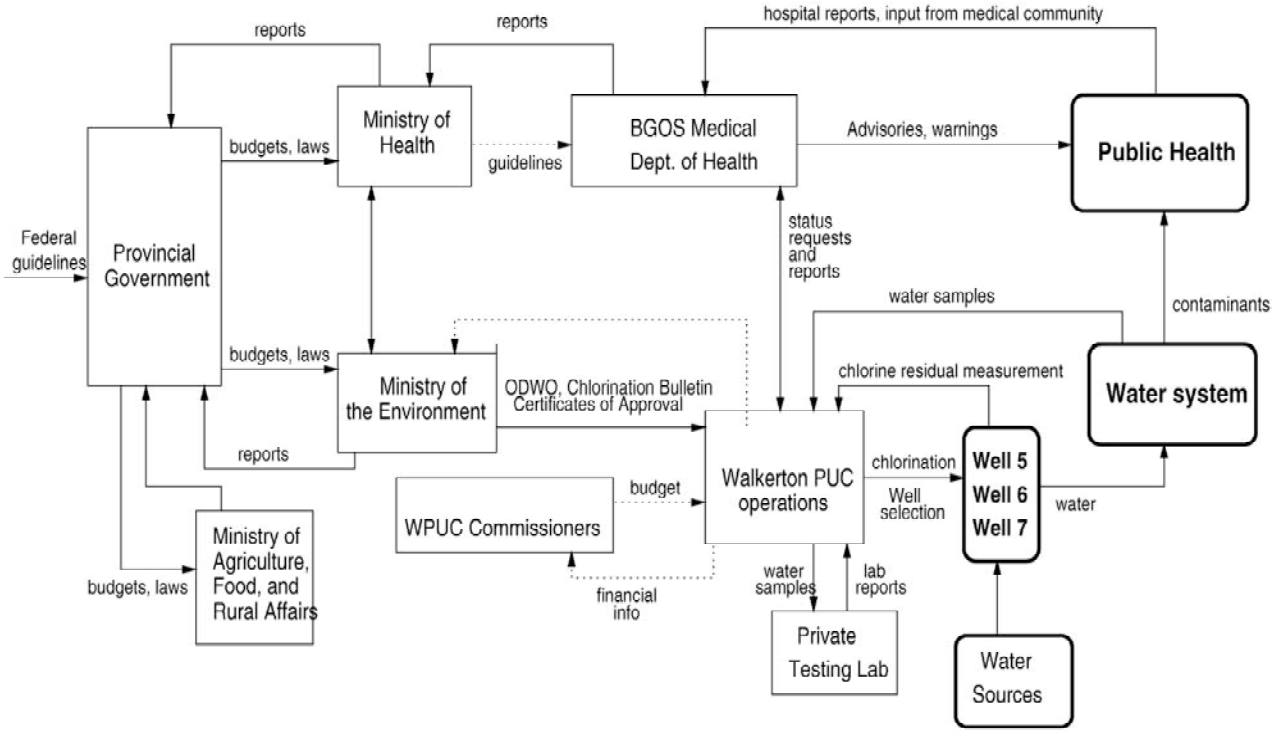


Figure 15 – The Basic Water Safety Control Structure at the Time of the Accident. Dotted lines denote communication, control or feedback channels that had become ineffective.

Proximate Events to the Accident: We now get to the proximate events preceding the accident—the place where many accident investigations begin and the start of accident models based on chains of failure events—and can see how they combined with the inadequate control structure in place at the time (Figure 15) to lead to the losses.

The source of the water contamination was manure that had been spread on a farm near Well 5. Unusually heavy rains from May 8 to May 12 carried the bacteria to the well. Between May 13 and May 15, Frank Koebel checked Well 5 but did not take measurements of chlorine residuals (*incorrect actuator operation*), although daily checks were supposed to be made. Well 5 was turned off on May 15.

On the morning of May 15, Stan Koebel returned to work after having been away from Walkerton for more than a week. He turned on Well 7, but shortly after doing so, he learned a new chlorinator for Well 7 had not been installed and the well was therefore pumping unchlorinated water directly into the distribution system (*incorrect control action*). He did not turn off the well, but instead allowed it to operate without chlorination until noon on Friday May 19, when the new chlorinator was installed (*incorrect control action*).

On May 15, samples from the Walkerton water distribution system were sent to A&L Labs for testing according to the normal procedure. On May 17, A&L Labs advised Stan Koebel that samples from May 15 tested positive for E. coli and total coliforms. The next day (May 18) the first symptoms of widespread illness appeared in the community. Public inquiries about the water prompted assurances by Stan Koebel that the water was safe (*incorrect feedback*). By May 19 the scope of the outbreak had grown, and a pediatrician contacted the local health unit with a suspicion that she was seeing patients with symptoms of E. coli.

The Bruce-Grey-Owen Sound (BGOS) Medical Department of Health (the government unit responsible for public health in the area) began an investigation. In two separate calls placed to Stan Koebel, the health officials were told that the water was "okay" (*incorrect feedback*). At that time, Stan Koebel did not disclose the lab results from May 15, but he did start to flush and superchlorinate the system to try to destroy any contaminants in the water. The chlorine residuals began to recover. Apparently, Mr. Koebel did not disclose the lab results for a combination of two reasons: he did not want to reveal the unsafe practices he had engaged in from May 15–17 (i.e., running Well 7 without chlorination), and he did not understand the serious and potentially fatal consequences of the presence of E. coli in the water system (*incorrect process model*). He continued to flush and superchlorinate the water through the following weekend, successfully increasing the chlorine residuals. Ironically, it was not the operation of Well 7 without a chlorinator that caused the contamination; the contamination instead entered the system through Well 5 from May 12 until it was shut down May 15.

On May 20, the first positive test for E. coli infection was reported and the BGOS Medical Department of Health called Stan Koebel twice to determine whether the infection might be

linked to the water system. Both times, Stan Koebel reported acceptable chlorine residuals and failed to disclose the adverse test results (*incorrect feedback*). The MDH assured the public that the water was safe based on the assurances of Mr. Koebel.

That same day, a WPUC employee placed an anonymous call to the Ministry of the Environment (MOE) Spills Action Center, which acts as an emergency call center, reporting the adverse test results from May 15. On contacting Mr. Koebel, the MOE was given an evasive answer and Mr. Koebel still did not reveal that contaminated samples had been found in the water distribution system (*inaccurate feedback*). The Local Medical Officer took over the investigation for the Medical Department of Health. He took independent water samples and delivered them to the Ministry of Health laboratory in London (Ontario) for microbiological testing.

When asked by the MOE for documentation, Stan Koebel finally produced the adverse test results from A&L Laboratory and the daily operating sheets for Wells 5 and 6, but said he could not produce the sheet for Well 7 until the next day. Later, he instructed his brother Frank to revise the Well 7 sheet with the intention of concealing the fact that Well 7 had operated without a chlorinator. On Tuesday May 23, Stan Koebel provided the altered daily operating sheet to the MOE (*inaccurate feedback*). That same day, the health unit learned that two of the water samples it had collected on May 21 had tested positive for E. coli.

Without waiting for its own samples to be returned, the BGOS health unit on May 21 had issued a boil water advisory on local radio (*only partially effective control action*). About half of Walkerton's residents became aware of the advisory on May 21, with some members of the public still drinking the Walkerton town water as late as May 23. The first person died on May 22, a second on May 23, and two more on May 24. During this time, many children became seriously ill and some victims will probably experience lasting damage to their kidneys as well as other long-term health effects and, as stated earlier, seven people died and more than 2300 became ill.

Looking only at these proximate events, this appears to be a clear case of incompetence, negligence, and dishonesty by WPUC employees. In fact, the government argued at the Inquiry that Stan Koebel and/or the Walkerton PUC were solely responsible for the outbreak and that they were the only ones who could have prevented it. The Inquiry board correctly ignored this argument, but in May 2003 (almost exactly three years after the accident), the Koebel brothers were arrested for their part in that accident. But a STAMP analysis (and the Inquiry report) provides a much more informative and useful understanding of the accident and what might be changed to prevent future repetitions besides simply firing the Koebel brothers or putting them in jail: The stage for the accident had been set over a large number of years by actions at all levels of the socio-technical system structure—an example of how complex socio-technical systems can migrate toward an accident.

If the control structure designed using STPA (or something similar) had been in effect in May 2000, a good argument could be made that the accident would not have occurred. Despite the government's argument that the accident was solely due to actions by the Koebels and the WPUC, the Inquiry report made many recommendations for changes to the public water safety control structure including establishing regulatory requirements for agricultural activities with

potential impacts on drinking water sources, updating of standards and technology, improving current practices in setting standards, establishing legally enforceable regulations rather than guidelines, requiring mandatory training for all water system operators and requiring grandparented operators to pass certification examinations within two years, developing a curriculum for operator training and mandatory training requirements specifically emphasizing water quality and safety issues, adopting a province-wide drinking water policy and a Safe Drinking Water Act, strictly enforcing drinking water regulations, and committing sufficient resources (financial and otherwise) to enable the MOE to play their role effectively. Most of these recommendations have not yet been implemented, and water contamination accidents have continued to occur in Ontario municipal water systems.

Conclusions and Future Directions

This report described a new approach to hazard analysis (STPA) based on the STAMP model of accident causation. STPA was evaluated by applying it to a complex collision avoidance system and the high-quality fault tree analysis that was performed for that system. The STPA analysis, while including everything in the fault tree analysis, was more comprehensive and complete. This report also showed how STPA can be applied to the operational, managerial, and regulatory components of an open system using a public water system as an example. While the hazard analysis of the collision avoidance system was performed for an existing system, the public water system example showed how STPA can be used throughout the system lifecycle to assist in risk-driven system design and performance monitoring of operations.

Because STPA uses executable and analyzable specifications/models, the potential arises for a risk-driven, model-based system engineering environment in which safety and other high-priority goals are designed into the system from the beginning. Much remains to be done, however, to make this vision into a reality. For example, one of the lessons we learned from performing the feasibility study in this SBIR research is the need for a format or notation for recording the information derived from STPA. We also need to determine how to design an interactive tool bench so the system engineer can perform STPA on the emerging system design. Such a tool bench might include tools for interactive control of simulation and analysis and for easily making changes in the models to investigate their implications for safety.

References

Bachelder, E. and Leveson, N.G. (2001) Describing and Probing Complex System Behavior: A Graphical Approach, 2001 SAE Transactions, Vol. 110, *Journal of Aerospace*, Section 1, pp. 263–273.

Benner, L. (1975) Accident Investigations: Multilinear Events Sequencing Methods, *Journal of Safety Research*, Vol. 7, No. 2, June, pp. 67-73.

Bureau of Air Safety Investigation (1996) Advanced Technology Aircraft Safety Survey Report. Dept. of Transportation and Regional Development, Australia, June.

Brehmer, B. (1992) Dynamic decision-making: Human control of complex systems. *Acta Psychologica*, Vol. 81, pp. 211-214.

Carroll, J. S. (1998) Organizational learning activities in high-hazard industries: The logics underlying self-analysis. *Journal of Management Studies*, 35, pp. 699-717.

Carroll, J. S., Rudolph, Jenny W., and Hatakenaka, Sachi (2002) Learning from experience in high-hazard organizations, submitted for publication.

Checkland, P. (1981) *Systems Thinking, Systems Practice*, John Wiley & Sons, New York.

Conant, R.C. and Ashby, W.R. (1970) Every good regulator of a system must be a model of that system. *International Journal of System Science*, 1, pp. 89-97.

Dulac, N., Viguier, T., Leveson, N., and Storey, M-A. (2002) On the Use of Visualization in Formal Requirements Specification, *International Conference on Requirements Engin*eering, Essen, Germany, September.

Edwards, W. (1962) Dynamic decision theory and probabilistic information processing, *Human Factors*, 4, pp. 59-73.

Fujita, Y. (1991) What shapes operator performance? *JAERI Human Factors Meeting*, Tokyo, November.

Jaffe, M.S, Leveson, N.G., Heimdahl, M.P.E., and Melhart, B.E. (1991) Software requirements analysis for real-time process-control systems, *IEEE Transactions on Software Engineering*, SE-17(3):241--258, March.

Johnson, W.G. (1980) *MORT Safety Assurance System*, Marcel Dekker, New York.

Klein, G.A., Orasano, R., Calderwood, R., and Zsambok, C.E. (eds.) *Decision Making in Action: Models and Methods*, Ablex Publishers, 1993.

Leplat, J. (1987). Occupational accident research and systems approach. In Jens Rasmussen, Keith Duncan, and Jacques Leplat (eds.) *New Technology and Human Error*, pp. 181-191, John Wiley.

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

Leveson, N.G., Reese, J.D., Koga, S., Pinnel, L.D., and Sandys, S.D. (1997) Analyzing Requirements Specifications for Mode Confusion Errors, *First International Workshop on Human Error and System Development*, Glasgow.

Leveson, N.G. (2003) A New Accident Model for Engineering Safer Systems*, Safety Science*, in press.

Leveson, N.G. (2000) Intent specifications: An approach to building human-centered specifications, *IEEE Trans. on Software Engineering*, January.

Leveson, N.G., Daouk, M., Dulac, N., and Marais, K. (2003) Applying STAMP in Accident Analysis, *Workshop on the Investigation and Reporting of Accidents*, September.

Marais, K. and Leveson, N.G. (2003) Archetypes for Organizational Safety, Workshop on the Investigation and Reporting of Accidents, Williamsburg, September.

Miles, R.F. (1973) *Systems Concepts: Lectures on Contemporary Approaches to Systems*, John Wiley & Sons, New York.

Perrow, C. (1984) *Normal Accidents: Living with High-Risk Technology*, Basic Books, New York.

Rasmussen, J. (1990) Human error and the problem of causality in analysis of accidents. In D.E. Broadbent, J. Reason, and A. Baddeley (Eds.), *Human Factors in Hazardous Situations*, Clarendon Press, Oxford, pp. 1-12.

Rasmussen, J., Pejtersen, A.M., and Goodstein, L.P. (1994*) Cognitive System Engineering*, John Wiley & Sons.

Rasmussen, J. (1997) Risk Management in a Dynamic Society: A Modeling Problem. *Safety Science*, vol. 27, No. 2/3, Elsevier Science Ltd., pp. 183-213.

Reason, J. (1990*) Human Error*, Cambridge University Press.

Senge, P.M (1990). *The Fifth Discipline: The Art and Practice of the Learning Organization*, Doubleday Currency, New York.

Sterman, J.D. (2000) *Business Dynamics: Systems Thinking and Modeling for a Complex World*, Mc-Graw Hill/Irwin.

Svedung, I. and Rasmussen, J. (2002) Graphic Representation of Accident Scenarios: Mapping System Structure and the Causation of Accidents, *Safety Science*, vol. 40, Elsevier Science Ltd., pages 397-417.

Vicente, K.J. (1995) A Field Study of Operator Cognitive Monitoring at Pickering Nuclear Generating Station, Technical Report CEL 9504, University of Toronto.

Weiss, K.A., Ong, E.C., and Leveson, N.G. (2003) Reusable Software Architectures for Aerospace Systems, *Aircraft Engineering and Aerospace Technology*, in press.

Woods, D.D. (1984) Some results on operator performance in emergency events. In D. Whitfield (ed.), *Ergonomic Problems in Process Operations*, Inst. Of Chemical Engineering Symp., 1984.

Woods, D.C. (1984) Some Results on Operator Performance in Emergency Events, *Ergonomic Problems in Process Operations*, Institute of Chemical Engineering Symposium.

Woods, D.D. (2000) Lessons from beyond human error: Designing for resilience in the face of change and surprise. *Design for Safety Workshop*, NASA Ames Research Center, October.

Zimmerman, M., Leveson, N.G., and Lundqvist, K. (2002) Investigating the Readability of State-Based Formal Requirements Specification Languages, *Int. Conference on Software Engineering*, Orlando, May.

Zsambok, C.E. and Klein, G. (1997) *Naturalistic Decision Making*, Lawrence Erlbaum Associates, 1997.

Appendix: An Introduction to SpecTRM-RL

The SpecTRM-RL language includes a graphical overview of the system structure along with specification of output messages, inputs, state variables, macros, and functions. The rest of this section describes each of these features and can be skipped by readers familiar with the language.

Graphical Specification of the System Model: Figure 16 shows the four main components of a SpecTRM-RL specification: (1) a specification of the supervisory modes of the controller being modeled, (2) a specification of its control modes (3) a model of the controlled process (or *plant* in control theory terminology) that includes the inferred operating modes and system state (these are inferred from the measured inputs), and (4) a specification of the inputs and outputs to the controller. The graphical notation mimics the typical engineering drawing of a control loop.

Every automated controller has at least two interfaces: one with the supervisor(s) that issues instructions to the automated controller (the supervisory interface) and one with each controlled system component (controlled system interface). The supervisory interface is shown to the left of the main controller model while the interface with the controlled component is shown to the right.

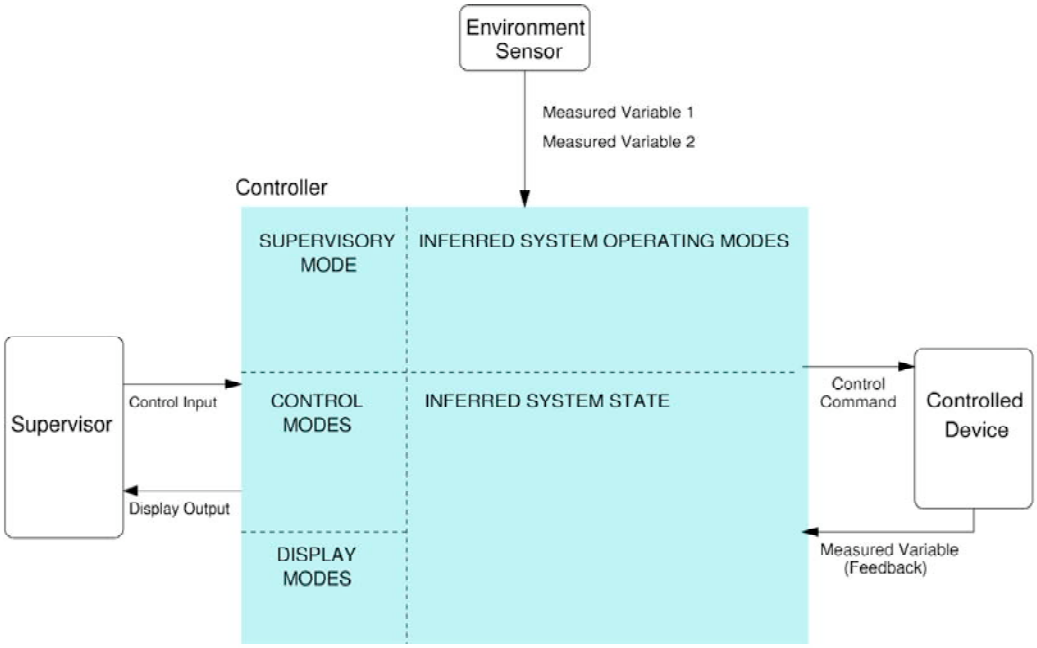


Figure 16 – The Parts of a SpecTRM-RL Graphical Model

The supervisory interface consists of a model of the operator controls and a model of the displays or other means of communication by which the component relays information to the supervisor. Note that the interface models are simply the logical view that the controller has of the interfaces the real state of the interface may be inconsistent with the assumed state due to

various types of design flaws or failures. By separating the assumed interface from the real interface, we are able to model and analyze the effects of various types of errors and failures (e.g., communication errors or display hardware failures). In addition, separating the physical design of the interface from the logical design (required content) will facilitate changes and allow parallel development of the software and the interface design. During development, mockups of the physical screen or interface design can be generated and tested using the output of the SpecTRM-RL simulator.

The bottom left quadrant of Figure 16 provides information about the control modes of the controller itself. These are not internal states of the controller (which are not included in our specifications) but simply represent externally visible behavior about the controller's modes of operation (described further below).

The right half of the controller model represents inferred information about the operating modes and states of the controlled system (the *plant* in control theory terminology). The model for a simple plant like a thermostat might include only one or two variables while that for a more complex system, e.g., air traffic control, might contain a large number of variables and include operational modes and models of multiple subcomponents. In a hierarchical control system, the controlled process may itself be a controller of another process. For example, the flight management system may be controlled by a pilot and may issue commands to a flight control computer, which issues commands to an engine controller. Parts of a SpecTRM-RL model can be reused or changed to represent different members of a product family [Weiss, Ong, and Leveson 2003].

Figure 17 shows the graphical part of a SpecTRM-RL specification of a simple altitude switch. The specification is based on an unpublished specification of an altitude switch by Steve Miller at Rockwell Collins. This switch turns on a Device of Interest (DOI) when the aircraft descends through a threshold altitude.

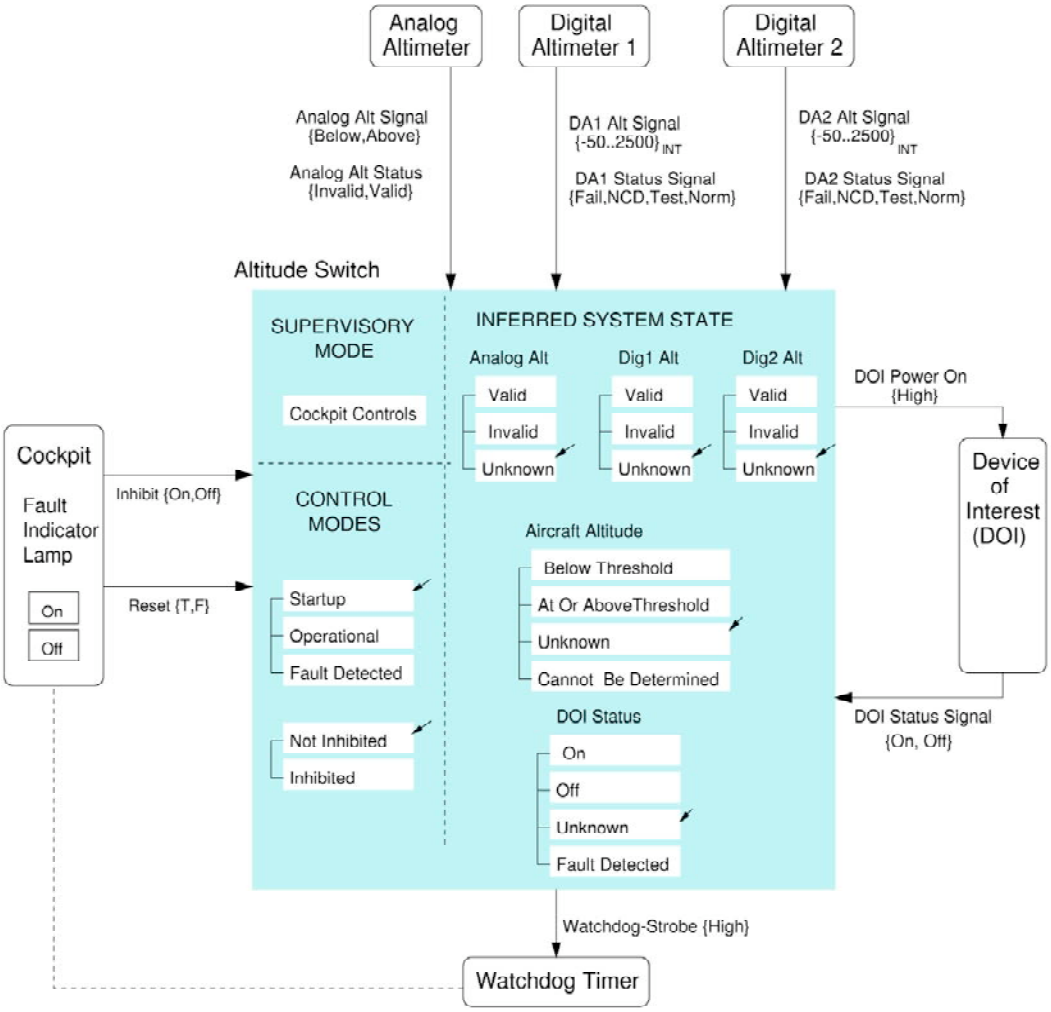


Figure 17 – Example of a Graphical Model

In SpecTRM-RL, state values in square boxes in the right side of the diagram represent inferred values used in the control of the computation of the blackbox I/O function. Such variables are necessarily discrete in value[4], and thus can be represented as a state variable with a finite number of possible values. In practice, such state variables almost always have only a few relevant values (e.g., altitude below a threshold, altitude at or above a threshold, cannot-be-determined, and unknown). Values for state variables in the plant model are required in SpecTRM-RL to include an *unknown* value. The meaning and purpose of the unknown state value are described below.

[4] If they are not discrete, then they are not used in the control of the function computation, but in the computation itself and can simply be represented in the specification by arithmetic expressions involving input variables.

In the altitude switch example, defining the control algorithm requires using information about the aircraft altitude level with respect to a given threshold, the inferred status of the DOI, and the validity of the altimeter information being provided as well as the measured variables and various constants defined elsewhere in the specification.

The possible values for a state variable are shown with a line connecting the boxes. The line simply denotes that the values are disjoint, that is, the variable may assume only one value at a time. A small arrow pointing at a box denotes the default (startup) value for the state variable or mode. For example, the DOI-Status can have the values *On, Off, Unknown*, and *Fault-Detected*. The default value is *Unknown*.

The altitude switch has two control inputs (shown on arrows to the left of the Component diagram): a reset signal that has the value *true* or *false* and an inhibit button that inhibits operation of the altitude switch. The inhibit button can either be in the *on* or *off* position. The only display in the altitude switch example is a fault indicator lamp that can also either be *on* or *off*, but its content is controlled through the watchdog timer and not directly by the altitude switch. There is only one supervisory mode cockpit controlled which is shown in the upper left quadrant of the component model.

Inputs representing the state of the plant (monitored or measured variables) are shown with arrows pointing to the controller. For the altitude switch, these variables provide (1) the current status (on or off) of the device of interest (DOI) that the altitude switch turns on and (2) inputs about the status and value of three altimeters on the aircraft (one analog and two digital) that provide information to the altitude switch about the current measured altitude of the aircraft as well as the status of that information (i.e., normal operation, test data, no computed data provided, or failed).

The output commands are denoted by outward pointing arrows. In the example, they include a signal to *power-on* the device (DOI) and a *strob*e to a watchdog timer so that proper action can be taken (by another system component) if the altitude switch fails. The outputs in this example are simple "high" signals on a wire or line to the device.

Note that the internal design of the altitude switch is not included in the model. The altitude switch operating modes are externally visible (and must be known for the pilot to understand its operation) and the aircraft model is used to describe the externally visible behavior of the altitude switch in terms of the process being controlled (and not in terms of its own internal data structures and algorithms). Thus the specification is blackbox.

Because of the simplicity of the altitude switch example, there are a few features of SpecTRM-RL that are not needed or are not well illustrated. Almost all of the missing features involve the ability to specify modes. Modes are abstractions on states and are not necessary for defining blackbox behavior. They are useful, however, in understanding or explaining the behavior of complex systems. While some formal specification languages use the term "mode" as a synonym for state (all modes are states and vice versa), SpecTRM-RL uses the more limited definition of mode common in engineering, i.e., as a state variable that plays a particular role in the state machine. In this usage, modes partition the state space into disjoint sets of states. For example,

the state machine may be in normal operational mode or in a maintenance mode. Our definition was chosen to assist in reviewability of the specification by domain experts and in formal analysis of specifications for particular properties commonly involved in operator mode confusion [Leveson et.al 1997]. SpecTRM-RL allows specifying several types of modes: supervisory modes, control modes, controlled-system operating modes, and display modes.

Supervisory modes are useful when a component may have multiple supervisors at any time. For example, a flight control computer in an aircraft may get inputs from the flight management computer and also directly from the pilot. Required behavior may differ depending on which supervisory mode is currently in effect. Mode-awareness errors related to confusion in coordination between multiple supervisors can be defined (and the potential for such errors theoretically identified from the models) in terms of these supervisory modes.

Control Modes control the behavior of the controller itself. Modern avionics systems may have dozens of modes. Control modes may be used in the interpretation of the component's interfaces or to describe the component's required process-control behavior. In the altitude switch, two types of control modes are useful in specifying the blackbox behavior: (1) whether the switch is in the *startup*, *operational*, and *internal-fault-detected* mode (the latter will result in the fault indicator light being lit in the cockpit and cessation of activity until the reset button is pushed) and (2) whether the operation of the altitude switch is partially inhibited or not. These two sets of modes cannot be combined in this case, as they are not disjoint. In fact, in our original specification of the altitude switch, they were combined. We later found that error through the use of our completeness criteria.

A third type of mode, *controlled-system* or *plant operating modes*, can be used to specify sets of related behaviors of the controlled-system (plant) model. They are used to indicate its operational status. For example, it may be helpful to define the operational state of an aircraft in terms of it being in takeoff, climb, cruise, descent, or landing mode. Such operating modes are not needed to define the behavior of the altitude switch and thus are not included in the example. In systems with complex displays (such as Air Traffic Control systems), it may also be useful to define various *display modes*.

Output Message Specification: Everything starts from outputs in SpecTRM-RL. By starting from the output specification, the specification reader can determine what inputs trigger that output and the relationship between the inputs and outputs. This relationship is the most critical in understanding and reviewing a system requirements specification, and therefore saliency of this information can assist in these tasks. Other state-machine specification languages, such as RSML and Statecharts, do not explicitly show this relationship, although it can be determined, with some effort, by examining the specification.

An example output specification is shown in Figure 18. The information included is influenced by the completeness criteria we previously defined for safety-critical blackbox specifications [Jaffe et. al 1991, Leveson 1995]. The completeness criteria also play an important role in the hazard analysis process described in Chapters 3 and 5.

The following information can and should be included in the specification of the output message: **destination** of the output; **acceptable values**; **timing behavior** including any initiation delay or completion deadline along with any required exception-handling behavior if the deadlines cannot be met, output load and capacity limitations, etc.; **feedback information** about how the controller will determine that the output command has been successfully implemented; and the identity of any other output commands that **reverse** this output.

In many systems, it is important to indicate a maximum time in which the output command remains effective before it is executed. After this time, the output essentially ``times out'' and should not be executed. This *data age* information can either be provided in the output message (if the timeout is implemented by the output command actuator) or included in the reversal information if the controller must issue a reversal to undo the command's effect. Reversal information is also useful in identifying accidentally omitted behavior from the specification, i.e., most output actions to provide a change in the controlled plant or supervisory interface have complementary actions to undo that change. **References** are pointers to other levels of the intent specification related to this part of the specification and are used for traceability.

Some of this information may not be applicable or important for a particular system. However, by including a place for it in the specification language syntax, the specifier must either include it or indicate that the information is not applicable or important. The implementers and maintainers need to know that these behaviors are not important and why not and also need to know that it was considered by the original specifiers and not simply forgotten. Lots of accidents and incidents result from such a lack of consideration of these factors by designers and implementers.

The conditions under which an output is triggered (sent) can be specified by a predicate logic statement over the various states, variables, and modes in the specification. In our experience in specifying complex systems, however, we found that the triggering conditions required to accurately capture the requirements are often extremely complex. We also found propositional logic notation did not scale well to complex expressions in terms of readability and error-proneness. To overcome this problem, we developed a tabular representation of disjunctive normal form (DNF) that we call AND/OR tables.

The far-left column of the AND/OR table lists the logical phrases of the predicate. Each of the other columns is a conjunction of those phrases and contains the logical values of the expressions. If one of the columns evaluates to *true*, then the entire table evaluates to *true*. A column evaluates to *true* if all of its elements match the truth values of the associated predicates. An asterisk denotes “don't care.”

Output Command

DOI Power On

Destination: DOI

Fields:
- **Name:** Command
  - **Type:** Discrete signal
  - **Acceptable Values:** {high}
  - **Units:**
  - **Granularity:**
  - **Hazardous Values:**
  - **Description:**
  - **Comments:**

Timing Behavior
- **Initiation Delay:** 0 milliseconds
- **Completion Deadline:** 50 milliseconds
- **Exception-Handling:** (What to do if cannot issue command within deadline time)

Feedback Information:
- **Variables:** DOI Status Signal
- **Values:** high (on)
- **Relationship:** Should be on if ASW sent signal to turn on
- **Min. time (latency):** 2 seconds
- **Max. time:** 4 seconds
- **Exception Handling:** DO Status changed to Fault Detected

Reversed By: Turned off by some other component or components. Do not know which ones.

Comments: I am assuming that if we do not know if the DOI is on, it is better to turn it on again, i.e., that the reason for the restriction is simple hysteresis and not possible damage to the device.

This product in the family will turn on the DOE only when the aircraft descends below the threshold altitude. Only this page needs to change for a product in the family that is triggered by rising above the threshold.

TRIGGERING CONDITION

Control Mode	Operational	T
	Not Inhibited	T
State Values	DOI Status in state On	F
	Altitude in state Below Threshhold	T
	Previous value of Altitude in state At Or Above Threshold	T

MESSAGE CONTENTS

Field	Value
Command	HIgh

Figure 18 – Example of an Output Specification

For SpecTRM-RL, we kept the very successful AND/OR tables, but made one addition based on human factors considerations. We now recommend that the output condition tables be organized into two parts: an upper part denoting the relevant control modes for that column and a lower

part describing any additional conditions for triggering the output. We have found that this separation assists in completeness checking, particularly when humans are writing and reviewing specifications. For completeness reasons, every output command column must include a reference to the control mode(s) under which the command is sent. It is assumed that if a particular mode is not specified, then the output cannot occur in that mode.

For the altitude switch *DOI-Power-On* output in the example shown in Figure 18, the command is triggered (sent) when all the following conditions are true: the altitude switch is in the operational and not-inhibited modes, the DOI is not on, the altitude is below the threshold, and the previous altitude was at or above the threshold (the requirements for the altitude switch say that if the switch is turned off while the aircraft is below the threshold altitude, the DOI is not powered on again until the aircraft goes above the threshold altitude and again passes down through it). The *Previous* built-in function, which is a feature of the underlying formal RSM model, allows referring to previous values of modes, state variables, inputs, and outputs. References to time are also allowed in the specification of trigger conditions. An example is shown later.

Input Definition: Our desire to enforce completeness in the language itself (to satisfy our completeness requirements) leads to language features that allow the inclusion of information (if relevant) about input arrival rates, exceptional-condition handling, data-age requirements, etc. No input data is good forever; after some point in time it becomes obsolete and should not be used. We provide a special value, *obsolete*, that an input variable assumes a specified time after the last value is received for that variable.

In the example shown in Figure 19, the specification states that the value comes from the *altitude* field in the DA1-message and is assigned when a message arrives. If no message has arrived in the past 2 seconds, the previous value is used. If the last message arrived more than 2 seconds before, the data is considered obsolete. The input variable also starts with the obsolete (undefined) value upon startup. Because of the similarity of the form of most input definitions, we may simplify the notation in the future.

When the controller has multiple supervisory modes, these must be specified to denote which inputs should be used at any particular time.

State Variable Definition: State variable values are inferred from the values of input variables or from other state variable values. Figure 20 shows a partial example of a state variable description for the altitude switch.

Input Value

DA1 Alt Signal

Source: Digital Altimeter 1

Type: integer

Possible Values (Expected Range): -20..2500

Exception-Handling: Values below -20 are treated as -20 and values above 2500 as 2500

Units: feet AGL

Granularity: 1 foot

Arrival Rate (Load): one per second average

Min-Time-Between-Inputs: 100 milliseconds
Max-Time-Between-Inputs: none

Obsolescence: 2 seconds

Exception-Handling: Assumes value Obsolete

Description:

Comments:

Appears in: Altitude

DEFINITION

= New Data for DA1 Alt Signal

DA1 Alt Signal was Received	T

= Previous Value of DA1 Alt Signal

DA1 Alt Signal was Received	F
Time Since DA1 Slt Signal was last received > 2 seconds	F

= Obsolete

DA1 Alt Signal was Received	F	*	*
Time Since DA1 Alt Signal was last received > 2 seconds	T	*	*
System Start	*	T	*
DA1 Alt Signal was Never Received	*	*	T

Figure 19 - Example of an Input Specification

State Value

Altitude

Obsolescence: 2 seconds

Exception-Handling: Because the altitude-status-signals change to obsolete after 2 seconds, altitude will change to Unknown if all input signals are lost for 2 seconds.

Description: When at least one altimeter reports an altitude below the threshold, then the aircraft is assumed to be below the threshold. ↑2.12.1

Comments:

Appears in: DOI Power On

DEFINITION

= Unknown

System Start	T	*	*
Reset is High	*	T	*
Analog ALT in state Unknown	*	*	T
Dig1-Alt in state Unknown	*	*	T
Dig2-Alt in state Unknown	*	*	T

The altitude is assumed to be unknown at startup, when the pilot issues a reset command, and when no recent input has come from any altimeter.

= Below threshold

Analog Valid And Below	T	*	*
Dig1 Valid And Below	*	T	*
Dig2 Valid And Below	*	*	T

At least one altimeter reports a valid altitude below the threshold..

= At Or Above Threshold

Analog Valid and Above	T	T	T	F	T	F	F
Dig1 Valid and Above	T	T	F	T	F	T	F
Dig2 Valid and Above	T	F	T	T	F	F	T

At least one altimeter reports a valid altitude above the threshold and none below.

= Cannot Be Determined

Analog Alt in state Invalid	T
Dig1 Alt in state Invalid	T
Dig2 Alt in state Invalid	T

No valid data is received from any altimeter (all report test or failed status).

Figure 20 - Example of an Inferred State Variable Specification

As stated earlier, all state variables that describe the process state should include an *unknown* value. *Unknown* is the default value upon startup or upon specific mode transitions (for example, after temporary shutdown of the computer). This feature is used to ensure consistency between the computer model of the process state and the real process state upon startup or after leaving control modes where processing of inputs has been interrupted. By making *unknown* the default state value and by assuming the *unknown* value upon changing to a control mode where normal input processing is interrupted (for example, a maintenance mode), the use of an *unknown* state value forces resynchronization of the model with the outside world after an interruption in processing inputs. Many accidents have been caused by the assumption that the process state does not change while the computer is idle or by incorrect assumptions about the initial value of state variables on startup or restart.

If a model of a supervisory display is included in the specification, *unknown* is used for state variables in the supervisory display model only if the state of the display can change independently of the software. Otherwise, such variables must specify an initial value (e.g., blank, zero, etc.) that should be sent when the computer is restarted.

Macros and Functions: *Macros*, although not strictly necessary, were added to the language for human factors considerations. They are simply named pieces of AND/OR tables that can be referenced from within another table. For example, the macro in Figure 21 is used in the definition of the state variable *altitude* in the altitude switch example. Its use simplifies the specification of altitude and thus makes it easier to understand while also simplifying making changes and enhancing specification reuse. Macros, for the most part, correspond to typical abstractions used by application experts in describing the requirements and therefore add to the understandability of the specification. In addition, we have found this feature convenient for expressing hierarchical abstraction and enhancing hierarchical review and understanding of the specification. For very complex models (e.g., a flight management system), we have found that macros are an important tool for humans to be able to handle the complexity involved in constructing the specification.

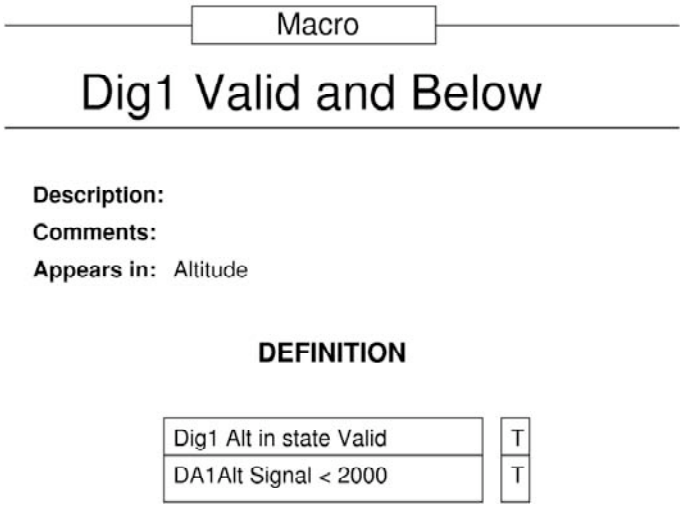


Figure 21 - Example of a Macro Specification

Rather than including complex mathematical functions directly in the transition tables, *function*s may be specified separately and referenced in the tables. In addition, functions are used in output specifications to define the value computed for the output (e.g., the differential equation used to define a control law).

The macros and functions, as well as other features of SpecTRM-RL, not only help structure a model for readability, they also help organize models to enable specification reuse. Conditions commonly used in the application domain can be captured in macros and common functions can be captured in reusable functions. Naturally, to accomplish reuse, care has to be taken when creating the original model to determine what parts are likely to change and to modularize these parts so that substitutions can be easily made.