Rasmussen's Legacy: A Paradigm Change in Engineering for Safety[1]

Nancy G. Leveson

Aeronautics and Astronautics Dept.

MIT

leveson@mit.edu

Introduction

New ideas in engineering often appear in multiple places around the same time when the world is ready for and needs them. A systems approach to safety appeared at the same time in several different contexts: Rasmussen in human factors and safety, von Bertalanfly (1968) in biology, and System Safety in the American ballistic missile program. The primary factor driving these ideas was greatly increased complexity in the systems we were building. The old approaches simply did not scale.

Rasmussen was trained in electrical and control engineering and much of his work reflects that engineering training. I believe his greatest achievements are in integrating human factors and engineering and applying the resulting ideas to safety. However, his ideas have had the most impact on human factors and cognitive systems engineering and not on the engineering of the hardware and software beyond the interface design. My goal has been to apply the ideas beyond human factors in order to influence the way that engineers approach the entire engineering process for complex, safety-critical sociotechnical systems.

For the most part, system safety engineering (beyond the physical interface design) has either ignored the human operators or has treated human factors as an independent topic that is an appendage or parallel to the system engineering process. System engineering concentrates on designing hardware and software to prevent accidents. When the inevitable accidents result from the lack of consideration of humans in system design, the losses are simply blamed on the human operators. Most of the time, in fact, operator error can be directly traced to flaws in the technical design of the system. We will make only limited progress in system safety until we fix enlarge the scope of system engineering. Rasmussen provided clues about how to accomplish this goal.

My own attempts to extend Rasmussen's ideas to engineering practice have involved improving engineering specifications, particularly the requirements engineering process; creating a new, more powerful, model of accident causation that better explains the cause of accidents in human-operated, software-intensive, sociotechnical systems; and creating new hazard analysis techniques that integrate humans into the generation of causal scenarios. I describe each of these briefly in this paper, but first it is necessary to understand some basic ideas in systems theory, which is what underlies Rasmussen's (and my) approach.

[1] A version of the paper will appear in a special issue of *Applied Ergonomics*, on the Legacy of Jens Rasmussen, 2016.

An Introduction to Systems Theory

Systems theory first arose in the 1940s and 1950s as a reaction to the changes in engineering that were starting to appear at that time. The most important change was in the complexity of the systems we were building. As complexity increased, traditional engineering approaches to system design and analysis became less and less effective.

Traditionally, science and engineering have coped with complexity in two ways: analytic reduction and statistical analysis. Analytic reduction goes back to the time of Descartes and is what most scientists and engineers are taught. The strategy is basically to handle complex systems by (1) dividing the system into distinct parts for analysis, (2) examining the parts separately, and later (3) combining the separate analysis results to provide results for the system as a whole. The physical aspects are divided into physical components, the functional aspects into functional components, and behavior is treated as distinct events over time.

The usefulness of analytic reduction rests on several assumptions: (1) Each component or subsystem operates independently, (2) the analysis results are not distorted when considering the components separately, (3) the components act the same when examined singly as when playing their part in the whole, and (4) the events are not subject to feedback loops and non-linear interactions.

Traditional approaches to safety engineering were created using this approach. Accidents are assumed to be caused by component failure so the analysis involves dividing the system into components and identifying chains of directly related physical or functional component failures that can lead to a loss. Failure events are assumed to be random with a constant failure rate (exponentially distributed) so that the probability of a loss can be derived. After the component failure scenarios are identified, engineers use fault tolerance or fail-safe design techniques to protect against the component failures identified in the accident scenarios. Humans and social factors were for the most part omitted from consideration.

This approach to safety made sense for the relatively simple, pure electro-mechanical systems of the past where system components could be effectively decoupled, allowing simple, direct interactions among components. System design errors could, for the most part, be identified and eliminated by testing and what remained after development were random hardware failures. Operational procedures could be completely specified and human error mostly involved skipping a step or performing a step incorrectly. Reliability and safety were, therefore, closely related in these relatively simple designs.

This situation is now changing. Software is becoming an integral part of most systems, and it allows enormously more complex systems to be constructed; humans are increasingly assuming supervisory roles over automation, which requires more cognitively complex human decision making; and accidents

are more often resulting from unsafe interactions among components and not just individual or multiple component failures. While software design errors may exist that result in the software not implementing the stated requirements, the role of software in accidents and safety-related incidents is much more likely to result from inadequate software requirements. The software can be perfectly reliable (will do the same thing continually given the same inputs), but may still be unsafe.

The problems are similar for human operators. Assumptions about the role of human operators in safety have always been oversimplified. Most (many?) human factors experts now accept the fact that behavior is affected by the context in which it occurs and humans do not "fail" in a random fashion (see, for example, Dekker (2006), Flach (1995), Norman (2002), Rasmussen (1997)).

The basic problem is complexity. Complexity has increased in current advanced engineering systems to the point where all the potential interactions among system components cannot be anticipated, identified, and guarded against in design and operations. *Component interaction accidents* (as opposed to *component failure accidents*) are occurring where no components have "failed" but a system design error resulted in losses caused by previously unidentified, unsafe component interactions and component requirements specification errors. Hazard analysis techniques based on reliability theory and assumptions that accidents are caused by component failures do not apply to component interaction accidents.

As a result of these changes, new types of accidents are occurring and, in particular, are resulting from new causal factors, such as mode confusion or requirements incompleteness flaws (often missing cases) where nothing "failed" but the problem was in the overall system design. System theory handles these types of design problems.

Systems theory was created as an alternative to analytic reduction (von Bertalanffy, 1965; Weiner, 1965; Ackoff, 1971; Checkland, 1981). It focuses on systems taken as a whole, not on the parts considered separately. Systems theory assumes that some properties of systems can only be treated adequately in their entirety, taking into account all facets and relating the social to the technical aspects [Ramo 1973]. These system properties derive from the relationships among the parts of systems: how the parts interact and fit together [Ackoff 1971]. Thus the systems approach concentrates on the analysis and design of the whole as distinct from the components or parts.

Some properties in complex systems are *emergent*, that is, they arise from the interactions among the components. Safety is an emergent property. Looking only at one part of a complex system, it is not possible to tell whether an accident will occur. The property of "safety" only makes sense when all the interactions among the system components are considered together. The concept of emergence gives rise to the often quoted basic systems theory principle that in complex systems "the whole is greater than the sum of the parts."

Figure 1 depicts a system or process made up of components (the shaded boxes). The components interact in both direct and indirect ways. Emergent system or process properties arise from these interactions.

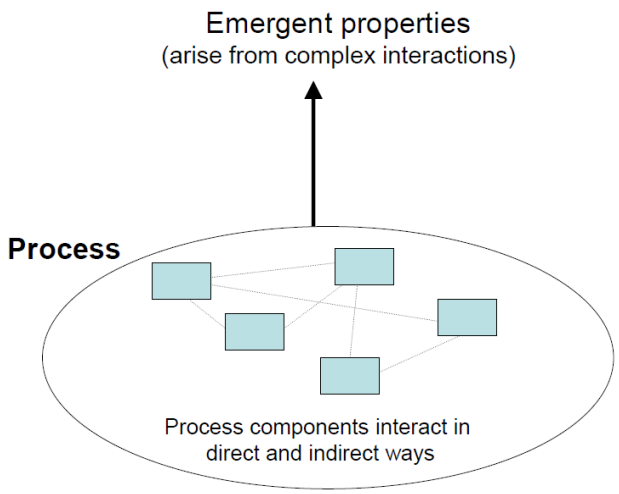

Figure 1: Emergent properties arise from component interactions

In order to ensure that the system is safe, constraints on the component interactions must be enforced. This need for enforcing safety constraints implies that there is some kind of "controller" that enforces the safety constraints by handling individual component behavior (including failures) and component interactions (Figure 2). Safety constraints are statements about what types of system level behavior or states are unacceptable, for example, power must never be on when the access door is open, two aircraft must not violate minimum separation requirements, the public must not be exposed to contaminated water and food products, airport runway incursions and operations on wrong runways must be prevented, or truck drivers must not drive when sleep deprived. Note the relationship between the safety constraints and the system hazards or risks.

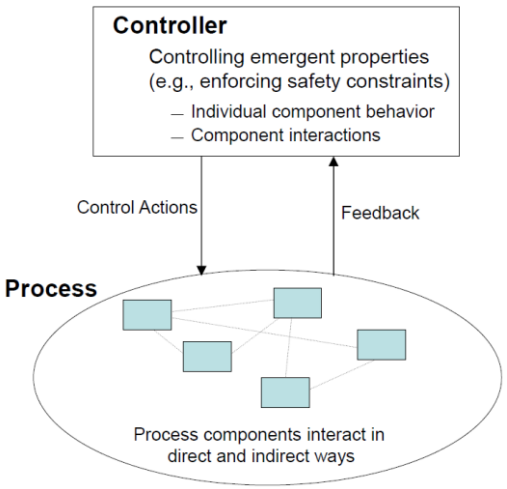

Figure 2: Safety constraints must be enforced on component interactions

The need for control does not imply there must be a physical human or automated controller to enforce constraints. A broad view of "control" is implied. For example, component failures and unsafe interactions may be controlled through design (e.g., standard redundancy, interlocks, and fail-safe design), or through processes (e.g., manufacturing or maintenance processes and operations) or through regulatory, legal, insurance, and other social controls. Control is also enforced indirectly by social and organizational culture: All behavior is influenced and at least partially "controlled" by the social and organizational context in which the behavior occurs such as policies, procedures, shared value systems (culture), and incentive structures and individual self-interest. Engineering this context can be an effective way to create and change a safety culture, i.e., the subset of organizational culture that reflects the general attitude about and approaches to safety by the participants in the organization or industry.

As an example, consider air traffic control. If each aircraft was allowed to optimize its own path without regard to what other aircraft were doing, the result would chaos and almost surely overall system throughput would be degraded. To avoid this, a central air traffic control system is used to control the flight paths of the aircraft in order to ensure that minimum separation between aircraft is maintained (and collisions do not result) and that the overall throughput is optimized even though each individual aircraft may not fly an optimal path for that aircraft.

An important result of this system's theoretic view of safety is that safety is treated as a *control* problem rather than a component reliability problem. Simply preventing component failures will not eliminate accidents caused by the unsafe interaction of components that have not failed.

Systems theory provided the theoretical and mathematical foundation for systems engineering, which was created in response to the problems that arose in designing the missile systems of the 1950s and 1960s. Complexity in these systems had reached the point where analytic reduction was no longer effective enough. New causes of accidents, related to the interaction among system components and not

necessarily component failures, started to occur. For that reason, a new approach to safety engineering based on systems theory was also created, called System Safety, although it had limited impact outside the U.S. defense community. Rasmussen's approach to safety clearly also was influenced and grounded in systems theory.

With this background, my implementation of Rasmussen's ideas in engineering can be understood. I have used his ideas in three ways: (1) creating new specification techniques using Rasmussen's Abstraction Hierarchy (AH), (2) creating a new, more powerful and inclusive model of accident causality and new hazard and risk analysis methods based on that model, and (3) augmenting hazard and risk analysis with consideration of how and why humans contribute to accidents.

System Development Specifications Based on Rasmussen's Abstraction Hierarchy

The human-machine interface provides a representation of the state of the system that the operator can use to solve problems and perform control, monitoring, and diagnosis tasks. Rasmussen's Abstraction Hierarchy (AH) has been used to create better human-machine interfaces by describing the system operator's functional work domain in a way that reflects the intentions and functional goals of the system operator. The AH represents constraints on the process, the interface, and the intent of the human operators of the process. The human-machine interface, in turn, provides a representation of the state of the system that the operator can use to solve problems and perform control, monitoring, and diagnosis tasks.

While much of the use of Rasmussen's AH has been in designing better interfaces between operators and automation and understanding work requirements, it is also a very useful approach to designing better engineering specifications that integrate operators into engineering design. Just as the control panel in a plant is the interface between the operator and the plant, system engineering specifications are the interface between the system customers or stakeholders, the designers and builders, and the maintainers. The specifications help the designer, implementer, tester, debugger, and maintainer understand the system requirements well enough to create a physical form, to find problems in that physical form, or to change the physical form. The ideas in Rasmussen's AH can to be used to specify and understand the relationship between the ends (goals) and the means (form) of the system design. The overall goal, as with the use of the AH in human-machine interfaces, is to represent the constraints on the design process. The result is called an *Intent Specification* (Leveson 2000).

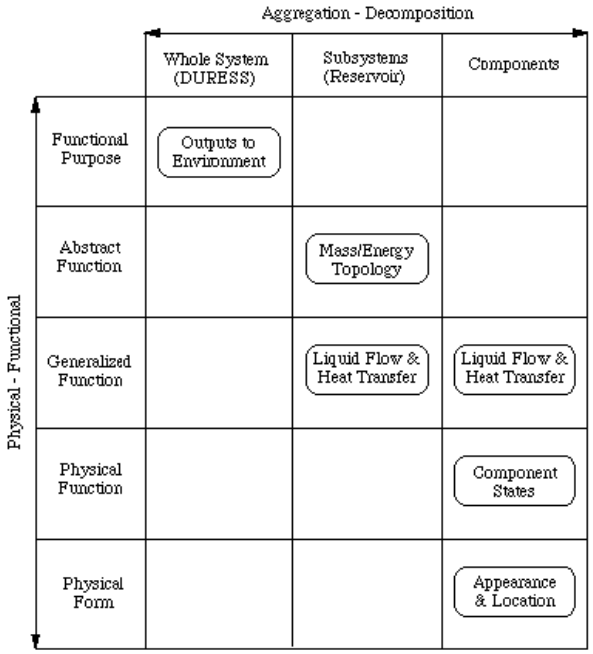

Figure 3: A Simple Example of the Abstraction Hierarchy

The goal of *Intent Specifications* is to enhance human processing and use by grounding specification design on psychological principles of how humans use specifications to solve problems as well as on basic system engineering principles. The language in which we specify problems has an effect on both our problem-solving ability and the errors we make while solving those problems. Cognitive psychology has firmly established that the representation of the problem provided to problem solvers can affect their performance (see Norman (1993) for a survey of this research). In fact, Woods (1995) claims that there are no neutral representations, that they either degrade or support performance. To provide assistance for problem solving, then, requires a theoretical basis for deciding which representations support effective problem-solving strategies. The goal of specification language design should be to make it easy for users to extract and focus on the important information for the specific task at hand without assuming particular mental models or limiting the problem-solving strategies employed by the users of the document. Rasmussen's AH provides part of that theoretical basis.

What types of specifications are needed to support humans in the system engineering process and to specify the results? Design decisions at each stage must be mapped into the goals and constraints they are derived to satisfy, with earlier decisions mapped (traced) to later stages of the process, resulting in a seamless (gapless) record of progression from high-level system requirements down to component requirements and designs. The specifications must also support the various types of formal and informal analyses used to decide between alternative designs and to verify the results of the design process. Specifications must in addition incorporate design rationale (intent) information. Finally, they must assist in the coordinated design of the components and the interfaces between them.

Rasmussen observed that the complexity of a system is not an objective feature of the system (Rasmussen, 1985). Observed complexity depends upon the level of resolution upon which the system is being considered. A simple object becomes complex if observed through a microscope. A way to cope with complex systems, therefore, is to structure the situation such that the observer can transfer the problem being solved to a level of abstraction with less resolution. "If a person is allowed to structure a complex system according to his perceptual and conceptual needs, sheer complexity is no bar to effective performance (Newman, 1966; Rasmussen, 1985).

Systems theory includes the concept of stratified hierarchies to deal with this complexity. Models of complex systems can be expressed in terms of a hierarchy of levels of organization, each more complex than the one below, where a level is characterized by having *emergent* properties. As noted in the previous section but stated here in a slightly different way, the concept of emergence is the idea that at any given level of complexity, some properties characteristic of that level (emergent at that level) are irreducible. Such properties do not exist at lower levels in the sense that they are meaningless in the language appropriate to those levels. For example, the shape of an apple, although eventually explainable in terms of the cells of the apple, has no meaning at that lower level of description. Figure 4 shows a model by Rasmussen (1997) of a stratified hierarchy of system operation related to safety.

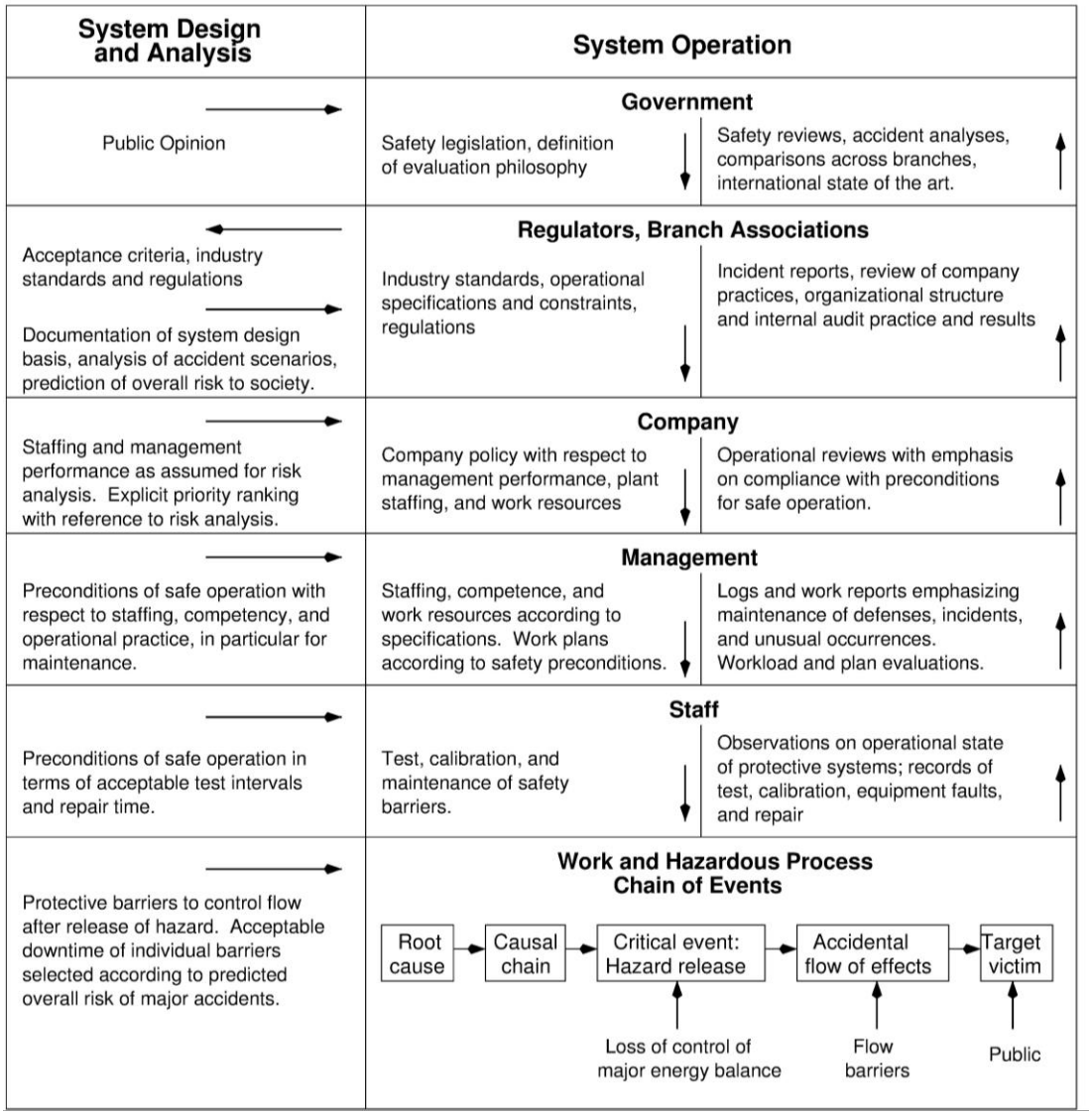

Figure 4: Hierarchical Model of Safety Control (Rasmussen, 1997)

Regulatory or control action, another basic concept of systems theory, involves imposing constraints upon the activity at one level of a hierarchy as described earlier. Those constraints define the "laws of behavior" at that level that yield activity meaningful at a higher level (emergent behavior). For example, the behavior at each level in Figure 4 affects whether the total system behavior is safe or not. Note also that describing emergent behavior or properties resulting from the imposition of constraints requires a language at a higher level (a metalevel) different than that describing the components themselves. Thus, different description languages are required at each hierarchical level.

Hierarchies are characterized by control processes operating at the interfaces between levels. The upper levels impose constraints on the lower ones through defined control actions. In Figure 4, the

downward arrows represent control actions on the level below and the upward arrows represent feedback about the state of the system below.

The basic concept of the AH forms the foundation for Intent Specifications. Engineers have traditionally used specifications based on decomposition and refinement. Decomposition was described earlier as analytic reduction. In contrast, refinement at each level implies that more detailed information is added than is available at the next higher level. Hierarchies based on decomposition and refinement, however, do not provide information about *why*. Higher-level emergent information about purpose or intent cannot be inferred from what we normally include in such specifications. Design errors may result when we either guess incorrectly about higher-level intent or omit it from our decision-making process. Rasmussen's AH provides a way to design engineering specifications that include such *why* information directly into the specification. Figure 5 shows the general form of an Intent Specification.

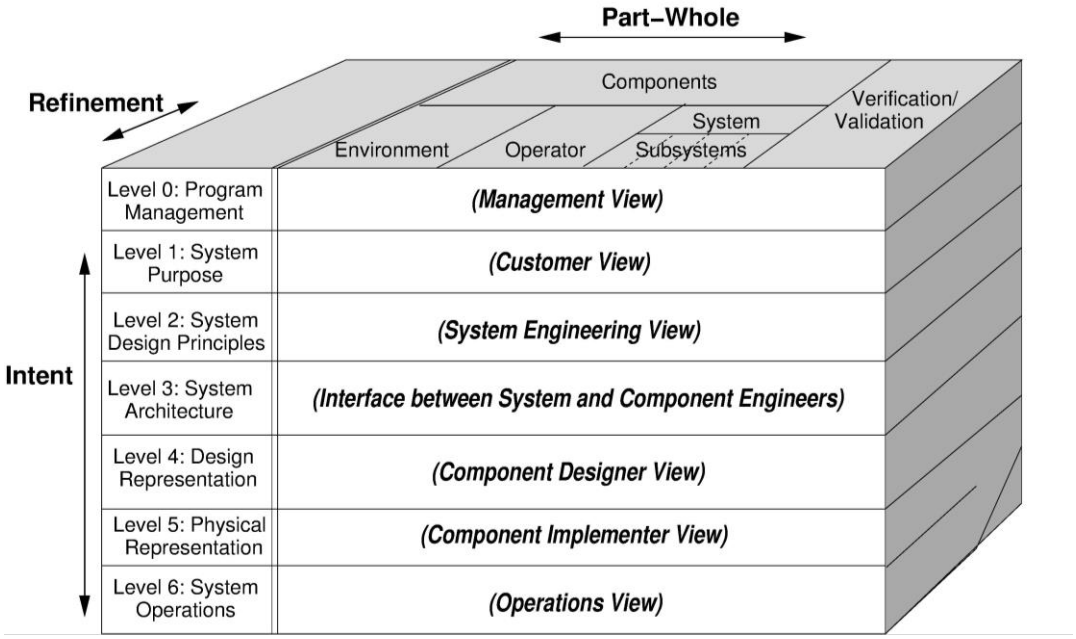

Figure 5: The Form of an Intent Specification

The design of Intent Specifications is based not only on the AH hierarchy, but also systems theory, system engineering principles, and psychological research on human problem solving and how to enhance it. The goal is to use these ideas to help engineers design and develop complex systems. It differs from a standard specification primarily in its structure, not its content: no extra information is involved that is not commonly found in detailed specifications—the information is simply organized in a way that has been found to assist in its location and use. Most complex systems have voluminous documentation, much of it redundant or inconsistent, and it degrades quickly as changes are made over time. Information about why something was done the way it was is usually missing, particularly design rationale. Trying to determine

whether a change might have a negative impact on safety, if possible at all, is usually enormously expensive and often involves regenerating analyses and work that was already done but either not recorded or not easily located when needed. Intent specifications embed design rationale, safety analysis results, and the assumptions upon which the system design and validation are based directly into the system specification and its structure, rather than being stored in separate documents, so information is at hand when needed for decision making.

Intent specifications are organized along three dimensions: intent abstraction, part-whole abstraction, and refinement (Figure 5). These dimensions represent the problem space in which the human navigates. Part-whole abstraction (along the horizontal dimension) and refinement (within each level) allow users to change their focus of attention to more or less detailed views within each level or model. The vertical dimension specifies the level of intent at which the problem is being considered.

Levels 1 through 5 represent the original five AH levels. A level (view) for management and one for system operations are added below and above, respectively, the original five levels. Each intent level contains information about the characteristics of the environment, human operators or users, the physical and functional system components, and requirements for and results of verification and validation activities for that level. Each level represents a different model of the system from a different perspective and supports a different type of reasoning about it. Refinement and decomposition occurs within each level of the specification rather than between levels. Each level provides information not just about what and how, but why, that is, the design rationale and reasons behind the design decisions, including safety considerations. Figure 6 shows an example of the information that might be contained in each level of the intent specification.

	Environment	Operator	System and components	V&V
Level 0 Prog. Mgmt.	Project management plans, status information, safety plan, etc.			
Level 1 System Purpose	Assumptions Constraints	Responsibilities Requirements I/F requirements	System goals, high-level requirements, design constraints, limitations	Preliminary Hazard Analysis, Reviews
Level 2 System Principles	External interfaces	Task analyses Task allocation Controls, displays	Logic principles, control laws, functional decomposition and allocation	Validation plan and results, System Hazard Analysis
Level 3 Blackbox Models	Environment models	Operator Task models HCI models	Blackbox functional models Interface specifications	Analysis plans and results, Subsystem Hazard Analysis
Level 4 Design Rep.		HCI design	Software and hardware design specs	Test plans and results
Level 5 Physical Rep.		GUI design, physical controls design	Software code, hardware assembly instructions	Test plans and results
Level 6 Operations	Audit procedures	Operator manuals Maintenance Training materials	Error reports, change requests, etc.	Performance monitoring and audits

Figure 6: Examples of the type of information that might be included in an Intent Specification

A New Systems-Theoretic Model of Accident Causation based on Systems Theory and Rasmussen's Interpretation of Systems Theory

Models of accident causality determine the way we try to understand and prevent accidents. The traditional model of accidents views them as a chain of component failure events, i.e., failure A causes failure B, which causes the loss event. This model implies that we can prevent accidents by either (1) preventing component failures or their effects, such as using high-integrity components or redundancy, or (2) preventing the propagation of failures by using barriers that prevent one failure event from leading to another one. An example of a barrier is a containment vessel in a nuclear power plant to contain any radioactivity that might result from a failure in the plant. Barriers, of course, are subject to failures themselves.

Systems theory provides a new theoretical framework for viewing accident causality and forms the basis of a new accident causality model called STAMP (System-Theoretic Accident Model and Processes) (Leveson, 2004; Leveson, 2012) that extends the view of causality as including more than component failures. Instead of focusing on preventing failures, STAMP shifts the focus to the larger issue of ensuring that safety constraints are enforced, which *includes* preventing those failures that will lead to a loss. Accidents result from a lack of enforcement of safety constraints on design and operations. It also views accidents as a complex process involving the entire sociotechnical system rather than just a simple chain

of events. As such, it is applicable to complex systems and more sophisticated views of human error than treating human error as simply a result of random "failure."

The STAMP model of causation is a direct translation of system theory to safety and has three primary components: safety constraints, hierarchical safety control structures, and process models. Safety constraints were described earlier.

Hierarchical Safety Control Structure

Rasmussen was the first to describe a safety control structure, shown in Figure 4. His emphasis was on operations, but safety also has to be designed into systems. An example of a more comprehensive hierarchical safety control structure is shown in Figure 7. Chains of events (as in the Rasmussen model in Figure 4) are not included at all as STAMP conceives accident causation as a more complex process than a simple chain of events.

There are two basic control structures in the example, one for operations (on the right) and one for development (on the left). The structures are connected through various interactions (only a few are shown to make the figure more readable). In the bottom right is the operating process. The physical process increasingly today has an automated controller along with human operators (controllers) overseeing the automation. Human operators may or may not have some limited direct (i.e., not computer-mediated) interactions with the physical process (shown as dotted lines). Actuators implement the commanded control actions, where actuators can be humans or machines, and sensors provide feedback about the state of the process.

Controllers are responsible for ensuring the enforcement of the safety constraints on the operating process. Above the operating process there is a safety management structure. For example, operations management provides work instructions (downward arrow) and gets feedback about the state of the process (upward arrow). Similarly, company management, regulators, insurance companies, the courts, legislatures, etc. all are part of the overall safety control structure.

Each component of this structure plays a role in and has responsibility for enforcing the safety constraints in the process they are controlling, although the assigned parts of the overall system safety constraints will probably differ for each component. Responsibilities will be more general and encompassing at the higher levels of the safety control structure while those at the lower levels will be more direct and specific. OSHA, for example, is responsible for ensuring that general safe workplace practices are being used throughout industry while a specific worker may be responsible for ensuring that a tank does not overflow and cause a dangerous spill of toxic chemicals in a particular plant.
Systems must also be designed and constructed to be safe, which is the role of the safety control structure on the left in Figure 3.

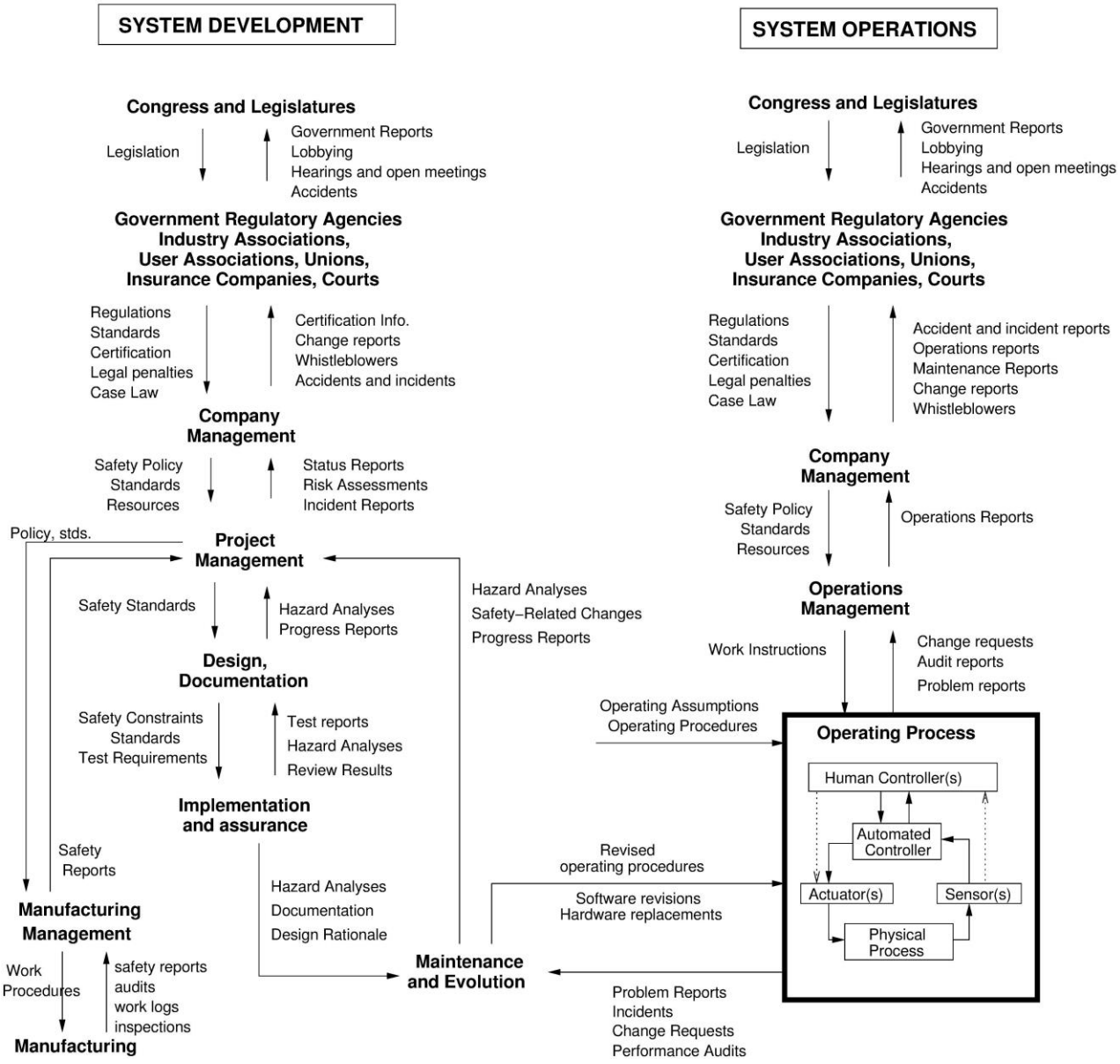

Figure 7: A Generic Hierarchical Safety Control Structure

Control structures can and will change over time. The goal is not to avoid change at all cost—change and flexibility are necessary for long-term organizational success—but to be cognizant of the potential influences of such changes and ensure that the safety constraints in place are not adversely impacted. Accidents often occur after changes. Management of Change (MOC) procedures are usually incorporated in company safety management systems. However, these policies are often not followed and, more important, only handle planned changes. Unplanned changes, such as changes in human behavior over time or changes in the environment, that can lead to increased risk must be monitored and handled also. STAMP-based tools can be used to identify this migration toward states of higher risk (Leveson, 2015).

Among the system factors most likely to cause systems to migrate toward high-risk or unsafe conditions are economic and political factors (Rasmussen, 1997). Important changes include changing organizational goals in response to changing markets, competitive pressures, and governmental oversight policies. Such pressures are often identified as underlying major accidents, such as those involving the Space Shuttles Challenger and Columbia, which involved budget pressures, the drive toward increased Shuttle launch frequency, pressures by international partners, etc. (Leveson, 2007). Safety culture is another important aspect of safety control structures. Safety culture is set by management and communicated to employees through components of the safety control structure. Thus safety culture is part and parcel of the hierarchical safety control structure.

Process Models

The third major component of STAMP is process models. Control processes operate between each of the levels of the safety control structure. Figure 8 shows a simple control loop. The controller issues control instructions/actions to the controlled process. Decisions about what types of actions are needed is based on control algorithms, procedures, and an internal model of the state of the controlled process called a process model or, in humans, more commonly referred to as a mental model. The process model is kept up to date by feedback from the controlled process.

Accidents often occur when the process model becomes inconsistent with the real state of the process, i.e., the operator thinks the machinery has been shut down (although it is still operational) and enters the area to do maintenance and is injured or the operator thinks that the tank is not full yet (while in actuality it is) and does not turn off the pump before an overflow occurs. Cognitive engineers give the label "situation awareness" to the state of the process model. If the process model is incorrect, the operator may issue a control action (or perform an activity) that is unsafe. The most common reason for incorrect process models is missing, incorrect, misinterpreted, or delayed feedback. Additional reasons may be related to incorrect assumptions or information about the environment in which the controller is operating.

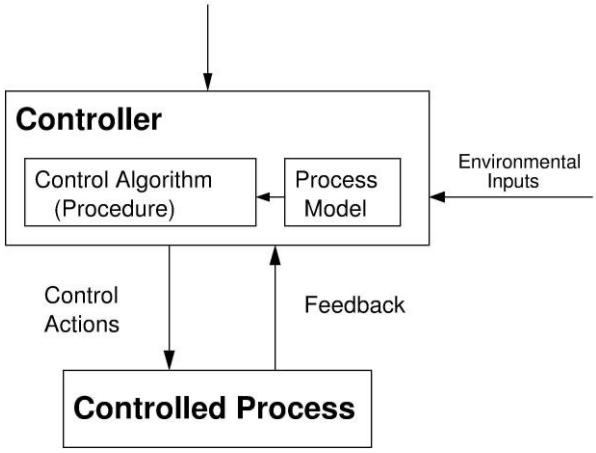

Figure 8: A Simple Control Loop Showing the Process Model

Note that the concept of process models is more useful to explain and handle human error than simply labeling it as "operator failure" or (random) human error. To improve safety, the information the operators need in order to make safe control decisions can be identified as well as the reasons why the process model may be inconsistent with the actual system state. That information can be used to design safer workplaces. We have developed methods to perform these types of analysis.

There are basically four types of unsafe control actions:

1. A control action required for safety is not provided, e.g., two aircraft are on a collision course but no advisory is provided to change their paths, equipment is not powered down before entering potentially dangerous areas or filling a tank is not stopped when the tank contents reaches the maximum limit. Another case is where a required safety control action is provided but is not followed. This latter case is the typical physical "failure scenario" in traditional accident models, e.g., a control action is sent to stop filling the tank but is not executed for some reason.
2. An unsafe control action is provided, e.g., an advisory is provided to aircraft that are not on a collision course but following the advisory puts them on one, an operator enters an area even though a lock-out/tag-out procedure has been performed, or an operator sends a control action to start filling a tank that is already full.
3. A potentially safe control action is provided too late, too early, or out of sequence, e.g., the driver hits the brakes but too late to avoid a collision or an advisory to prevent a collision between two aircraft is provided but too late to avoid the accident.
4. A safe control action is stopped too soon or applied too long. This type of factor applies only to continuous control actions, such as the operator not opening the valve far enough to allow an adequate flow of liquid through the pipe or a pilot stopping too soon when given an advisory to descend to a lower altitude in order to avoid a collision.

This model of causation provides a much deeper explanation for accidents involving software or human operators than a component failure model. Software and humans do not simply fail in a random fashion, as assumed by most traditional safety engineering approaches, but are influenced by their model of the controlled process and the environment (including cultural and organizational factors) in which they are working. They may also be given procedures that are inadequate or obsolete because of system changes of which they are unaware.

STAMP-Based Analysis Tools

Analysis methods and tools can be built on the STAMP foundation to assist in creating and operating complex, safety-critical systems. Such tools must consider a broad set of causes of accidents in complex, sociotechnical systems that derive from the basic STAMP model, such as inadequate technical system design, overlapping areas of responsibility and uncoordinated control actions, poor decision making due to inadequate process models, communication problems, weak or inadequate safety control structure design, etc.

CAST (Causal Analysis based on Systems Theory) is an accident analysis tool that can be used to identify the causal factors involved in an accident (Leveson, 2012). Typically, accident investigations focus on what failures led to the loss and who was responsible and should be blamed. CAST, in contrast, is used to determine why the safety control structure was not adequate to prevent the loss. Assigning blame is not the goal, that is, CAST is based on the assumption that the role of engineering is to determine *why* a loss occurred, not *who* was to blame.

CAST does not look for a single "root cause" or even several of them, but instead identifies the inadequacies in the safety control structure that allowed the events to occur. AcciMaps (Rasmussen and Svedung, 2000) also constructs a hierarchical control structure in analyzing accidents but CAST produces more details about the causes of the accident. Rather than stopping with identifying an operator or operators that did not provide safe control, it instead identifies why it made sense for the operators to act the way they did both in terms of the context of the events and any flaws in the process models that may have led to unsafe behavior. Recommendations can then be made to make the changes necessary to change the contextual factors that led to the unsafe control. The entire safety control structure (involved in the accident) is examined. Fully understanding the behavior at any level of the sociotechnical control structure requires understanding how and why the control at the next higher level allowed or contributed to the inadequate control at the current level.

STPA (System-Theoretic Process Analysis) is a proactive hazard analysis technique that essentially "investigates accidents before they occur." Causal scenarios are generated that *could* lead to accidents and the system design changed to eliminate or control them. Knowledge of what generally can go wrong in control loops is used in the analysis, as shown in Figure 9. STPA can be used early in system development to produce safer system designs from the beginning.

STAMP can also be the basis for procedures to create effective leading indicators that identify when risk is increasing and accidents are becoming more likely (Leveson, 2015). The ideas here are based on STAMP and on Rasmussen's theory of migration towards states of higher risk (Rasmussen, 1997).

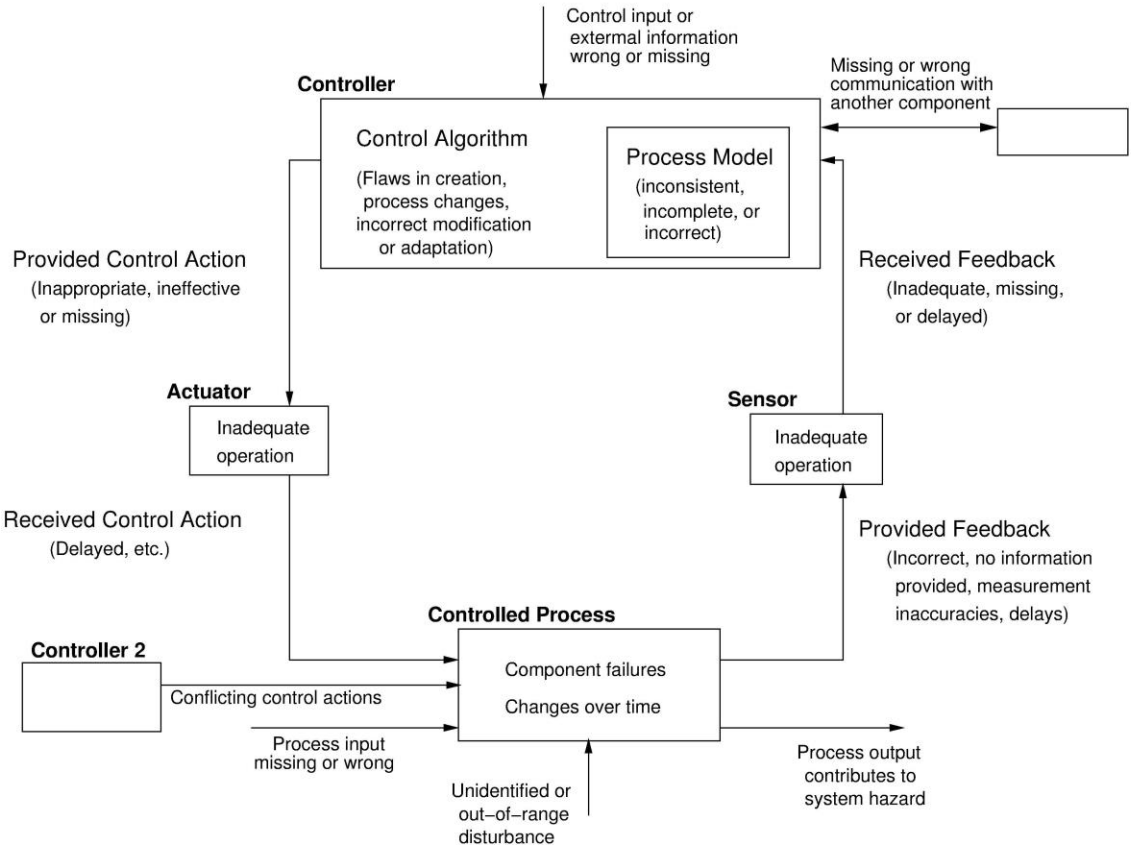

Figure 9: Some Causes of Inadequate Control in Control Loops

Including Human Factors in Hazard Analysis using Rasmussen's Model of Human-Task Mismatch

Engineers have typically ignored humans in hazard analysis or treat them as if they fail randomly like hardware devices. While the concept of a process/mental model in STAMP assists greatly in integrating humans into the engineering hazard analysis process, the model in Figure 8 is over simplified.

Figure 10 is a more realistic model. In the past, humans controlled physical devices directly, but today it is common for a computer or automated controller to provide direct control of the process and humans to control the automation. The human controller may only have indirect information about the controlled process. In addition to a model of the controlled process, the human controller must also have a model of

the state of the automation, the state of the controlled process, and the context in which decisions are being made.

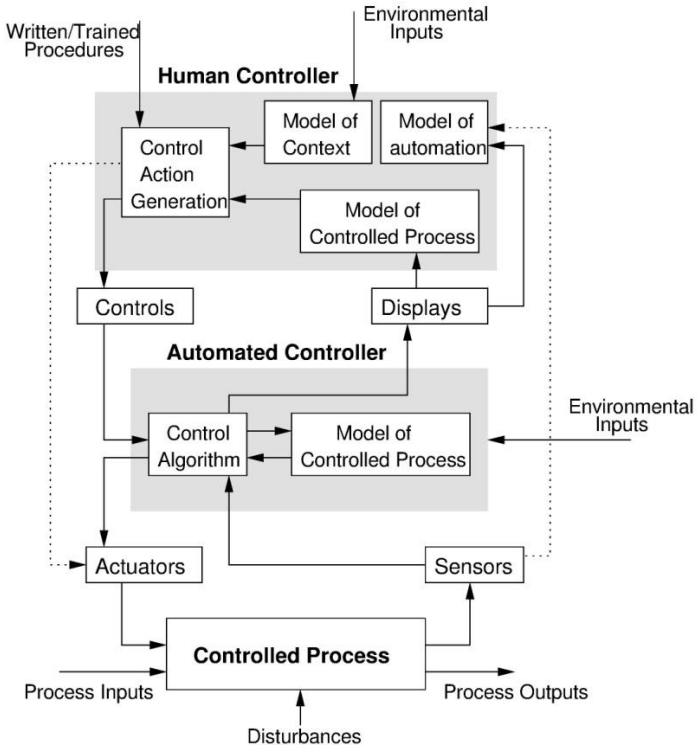

Figure 10: A model of a human controller, who controls an automated process, which is controlling a physical process

When humans controllers are involved (top box in Figure 10), context and behavior-shaping mechanisms play an important part in behavior and causality. The figure also shows a "control action generator" instead of a procedure or algorithm. Humans do not always follow procedures, nor should they. We use humans to control systems because of their flexibility and adaptability to changing conditions and ability to improvise when incorrect assumptions are made by the designers (Leveson, 2012).

While this model does allow us to perform more sophisticated analysis of the contribution of humans to accidents so that system designs can be improved, more is needed. We are currently working on improving the model of the human controller (top box in Figure 10) and augmenting our hazard analysis techniques. One possibility we are experimenting with is to use Rasmussen's model of human–task mismatch (Rasmussen, 1982).

Rasmussen argued that "errors" are an integral part of the learning process and should be regarded as human–task or human–system mismatch. He proposed a model, shown in Figure 11, based on the factors

necessary to explain the human behavior that precedes an accident. These factors are found by backtracking along the causal course of events, starting with the external model of malfunction and moving through the internal mode of malfunction, the mechanisms of human malfunction, and the causes of human malfunction. This model encompasses the more often cited Skills-Rules-Knowledge model.

This backtracking can be likened to backtracking through the parts of the control loop as done in STPA and CAST. Using the human–task mismatch model, the model in the top box of Figure 10 would look more like Figure 12.

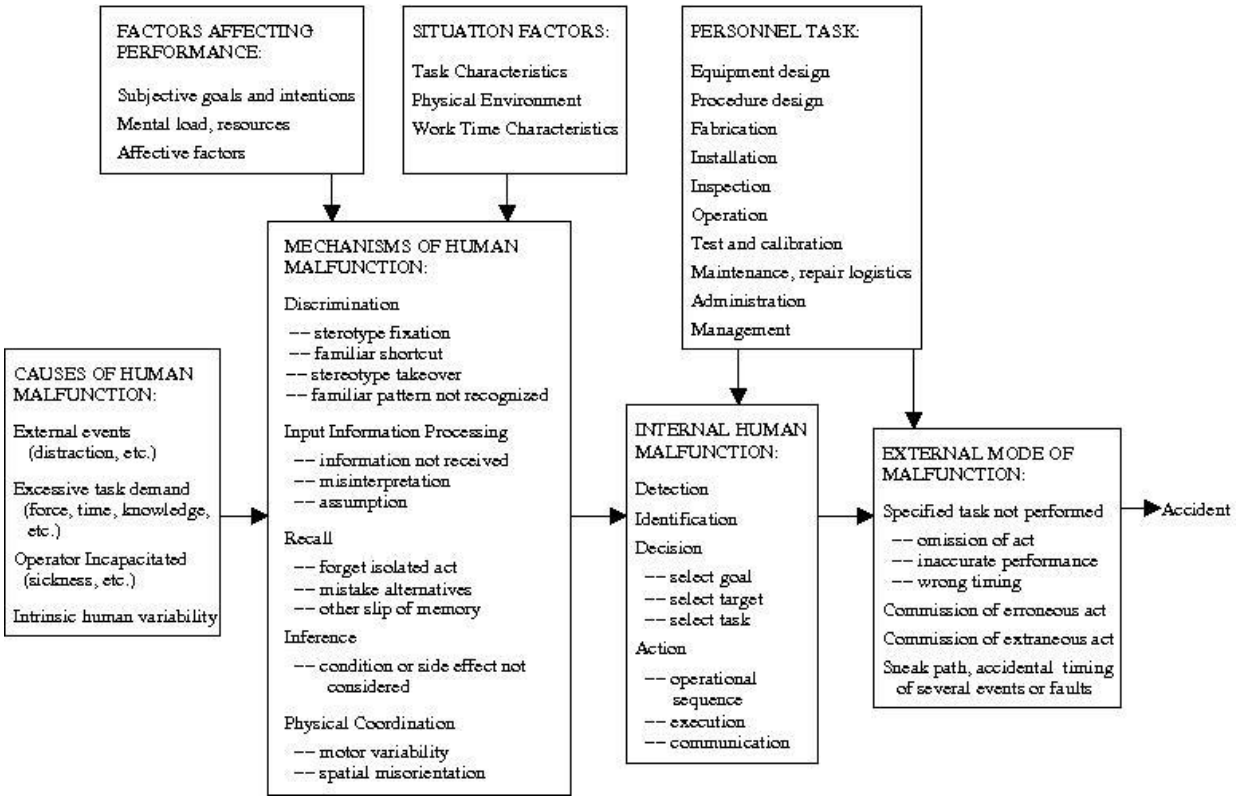

Figure 11: Rasmussen's Model of Human–Task Mismatch (Rasmussen, 1982)

.

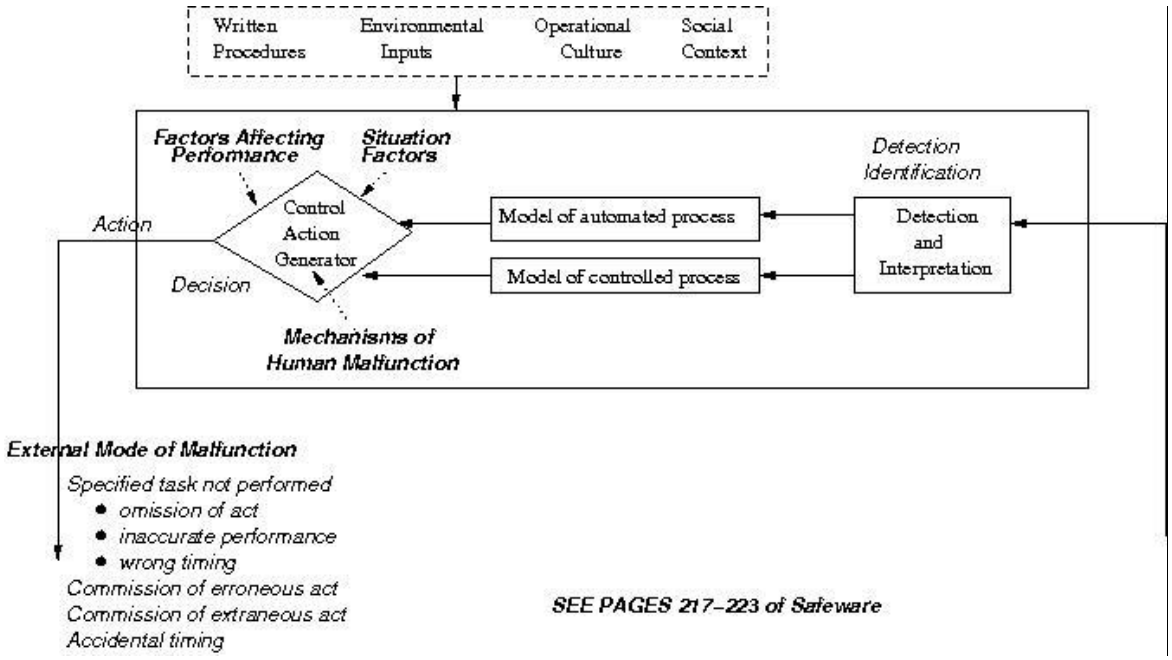

Figure 12: Rasmussen's Model of Human–Task Mismatch Integrated into the STAMP Process Model

Conclusions

Progress in engineering stems from building on the important ideas of the past. Rasmussen led the way in both understanding the importance of integrating human factors into engineering and providing a foundation for doing so. Those of us following in his footsteps are very grateful for his leadership.

References

Ackoff, R.L. (1971) Towards a system of systems concepts, *Management Science*, vol. 17, no. 11, pp. 661-671.

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

Dekker, S. (2006) *The Field Guide to Understanding Human Error*, London: Ashgate.

Flach, J., Hancock, P.A., Caird, J., and Vicente, K.J. (1995) *Global Perspectives on the Ecology of Human-Machine Systems*, CRC Press.

Leveson, N.G. (2000) Intent specifications: An approach to building human-centered specifications, *IEEE Transactions on Software Engineering*, vol. SE-26, no. 1.

Leveson, N.G. (2004) A New Accident Model for Engineering Safer Systems, *Safety Science*, vol. 42, no. 4, pp. 237-270.

Leveson, N.G. (2007) ``Technical and Managerial Factors in the NASA Challenger and Columbia Losses: Looking Forward to the Future,'' in Handelsman and Fleishman (eds.), *Controversies in Science and Technology, Vol. 2*: *From Chromosomes to the Cosmos*, Mary Ann Liebert, Inc.

Leveson, N.G. (2012) *Engineering a Safer World*, Cambridge, MA: MIT Press.

Leveson, N.G. (2015) A systems approach to risk management through leading safety indicators, *Reliability Engineering and System Safety*, Elsevier Scientific Publishers.

Newman, J.R. (1966) Extension of human capability through information processing and display systems, Technical Report SP-2560, Systems Development Corporation.

Norman, D.A. (1993) *Things that Make us Smart*, Addison-Wesley Publishing Company.

Norman, D. (2002) *The Design of Everyday Things*, New York: Basic Books

Ramo, S. (1973) The systems approach. In Ralph F. Miles Jr., editor, *Systems Concepts: Lectures on Contemporary Approaches to Systems*, pages 13–32, New York: John F. Wiley & Sons.

Rasmussen, J. (1982) Human errors: A taxonomy for describing human malfunction in industrial installations, *Journal of Occupational Accidents*, vol. 4, Elsevier Scientific Publishers, pp. 311-335.

Rasmussen, J. (1985) The role of hierarchical knowledge representation in decision making and system management, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-15, no. 2, March/April.

Rasmussen, J. (1986). *Information Processing and Human-Machine Interaction: An Approach to Cognitive Engineering*, North Holland.

Rasmussen, J. (1997) Risk management in a dynamic society: A modelling problem. *Safety Science*, vol. 27, no. 2/3, Elsevier Science Ltd., pp. 183-213.

Rasmussen, J. and Svedung, I. (2000), *Proactive Risk Management in a Dynamic So*ciety, Stockholm: Swedish Rescue Services Agency.

von Bertalanffy, L. (1968), *General Systems Theory: Foundations, Development, Applications*, Revised Edition (1976), New York: George Braziller.

Weiner, N. (1965) Cybernetics: or the Control and Communication in the Animal and the Machine, Cambridge, MA: MIT Press.

Woods, D.D. (1995), Toward a theoretical base for representation design in the computer medium: Ecological perception and aiding human cognition. In J.M. Flach, P.A. Hancock, K. Caird, and K.J. Vicente (editors), *An Ecological Approach to Human Machine Systems I: A Global Perspective*, Hillsdale, New Jersey: Erlbaum.