A Systems-Theoretic Approach to Safety in Software-Intensive Systems

Nancy G. Leveson

Abstract: Traditional accident models were devised to explain losses caused by failures of physical devices in relatively simple systems. They are less useful for explaining accidents in software-intensive systems and for non-technical aspects of safety such as organizational culture and human decision-making. This paper describes how systems theory can be used to form new accident models that better explain system accidents (accidents arising from the interactions among components rather than individual component failure), software-related accidents, and the role of human decision-making. Such models consider the social and technical aspects of systems as one integrated process and may be useful for other emergent system properties such as security. The loss of a Milstar satellite being launched by a Titan/Centaur launch vehicle is used as an illustration of the approach.

Keywords: software safety, system safety, accident models, software engineering

1 Introduction

All attempts to engineer safer systems rest upon underlying causal models of how accidents occur, although engineers may not be consciously aware of their use of such a model. An underlying assumption of these accident models is that there are common patterns in accidents and that accidents are not simply random events. By defining those assumed patterns, accident models may act as a filter and bias toward considering only certain events and conditions or they may expand consideration of factors often omitted. The completeness and accuracy of the model for the type of system being considered will be critical in how effective are the engineering approaches based on it.

At the foundation of almost all causal analysis for engineered systems today is a model of accidents that assumes they result from a chain (or tree) of failure events and human errors. The causal relationships between the events are direct and linear, 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 they cannot account for indirect and non-linear relationships.

The selection of events to include in an event chain is dependent on the stopping rule used to determine how far back the sequence of explanatory events goes. Although it is common to isolate one or more events or conditions (usually at the beginning of chain) and call them the *cause* or the proximate, direct or *root* cause of an accident or incident and to label the other events or conditions as *contributory*, there is no basis for this distinction. Usually a root cause selected from the chain of events has one or more of the following characteristics: (1) it represents a type of event that is familiar and thus easily acceptable as an explanation for the

accident; (2) it is a deviation from a standard; (3) it is the first event in the backward chain for which a "cure" is known[1]; and (4) it is politically acceptable as the identified cause. The backward chaining may also stop because the causal path disappears due to lack of information. Rasmussen suggests that a practical explanation for why actions by operators actively involved in the dynamic flow of events are so often identified as the cause of an accident (and operator actions are often selected as the stopping point in an accident event chain) is the difficulty in continuing the backtracking "through" a human [24]. Identifying accident causes in this way can be a hindrance in learning from and preventing future accidents.

As just one example, the accident report on a friendly fire shootdown of a helicopter over the Iraqi No-Fly-Zone in 1994 describes the accident as a chain of events leading to the shootdown [27]. Included in the chain of events provided is the fact that the helicopter pilots did not change to the radio frequency required in the No-Fly-Zone when they entered it (they stayed on the enroute frequency). Stopping at this event in the chain, it appears that the helicopter pilots were at least partially at fault for the loss by making an important mistake. An independent account of the accident [20], however, notes that the U.S. Commander of the operation had made an exception about the radio frequency to be used by the helicopters in order to mitigate a different safety concern, and therefore the pilots were simply following orders. This commanded exception to radio procedures is not included in the chain of events included in the official government accident report, but it provides a very different understanding of the role of the helicopter pilots in the loss.

There are two basic reasons for conducting an accident investigation: (1) to assign blame for the accident and (2) to understand why it happened so that future accidents can be prevented. When the goal is to assign blame, the backward chain of events considered often stops when someone or something appropriate to blame is found. As a result, the selected initiating event may provide too superficial an explanation of why the accident occurred to prevent similar losses in the future. For example, stopping at the O-ring failure in the *Challenger* accident and fixing that particular design flaw would not have eliminated the systemic flaws that could lead to accidents in the future. For *Challenger*, examples of those systemic problems include flawed decision making and the pressures that led to it, poor problem reporting, lack of trend analysis, a "silent" or ineffective safety program, communication problems, etc. None of these are "events" (although they may be manifested in particular events) and thus do not appear in the chain of events leading to the accident. Wisely, the authors of the *Challenger* accident report used an event chain only to identify the proximate physical cause and not the reasons those events occurred, and the report writers' recommendations led to many important changes at NASA or at least attempts to make such changes [26].[2]

Blame is not an engineering concept; it is a legal or moral one. Usually there is no objective criterion for distinguishing one factor or several factors from other factors that contribute to an accident. While lawyers and insurers recognize that many factors contribute to a loss event, for practical reasons and particularly for establishing liability, they often oversimplify the causes of accidents and identify what they call the *proximate* (immediate or direct) cause. The goal is to determine the parties in a dispute that have the legal liability to pay damages, which may be

[1] As an example, a NASA Procedures and Guidelines Document (NPG 8621 Draft 1) defined a root cause as: "Along a chain of events leading to a mishap, the first causal action or failure to act that could have been controlled systematically either by policy/practice/procedure or individual adherence to policy/practice/procedure."

[2] Recently, another Space Shuttle has been lost. While the proximate cause for the *Columbia* accident (foam hitting the wing of the orbiter) was very different than for *Challenger*, many of the systemic or root causes were similar and reflected either inadequate fixes of these factors after the *Challenger* accident or their re-emergence in the years between these losses [7].

affected by the ability to pay or by public policy considerations, such as discouraging company management or even an entire industry from acting in a particular way in the future.

When learning how to engineer safer systems is the goal rather than identifying who to punish and establishing liability, then the emphasis in accident analysis needs to shift from *cause* (in terms of events or errors), which has a limiting, blame orientation, to understanding accidents in terms of *reasons*, i.e., why the events and errors occurred. In an analysis by the author of recent aerospace accidents involving software in some way, most of the reports stopped after assigning blame—usually to the operators who interacted with the software—and never got to the root of why the accident occurred, e.g., why the operators made the errors they did and how to prevent such errors in the future (perhaps by changing the software) or why the software requirements specified unsafe behavior and why that error was introduced and why it was not detected and fixed before the software was used [12].

While attempts have been made to extend traditional safety engineering techniques such as fault tree analysis and probabilistic risk assessment, based on event-chain models of accidents to software-intensive systems, the results have not been terribly successful. Perhaps the lack of significant progress in dealing with software in safety-critical systems is the result of inappropriately attempting to extend the techniques that were successful in simpler, electromechanical systems and were based on models of accident causation that no longer apply.

Accidents can be separated into two types: those caused by failures of individual components and those caused by dysfunctional interactions between non-failed components. The dysfunctional behavior in modern, high-tech systems is often commanded by software, such as the command by the Mars Polar Lander descent control software to shut off the descent engines prematurely while still 40 meters above the Martian surface. In this and in most software-related accidents, the software operates exactly as specified, that is, the software, following its requirements, commands component behavior that violates system safety constraints or the software design contributes to unsafe behavior by human operators. As such, the traditional event-chain model, with its emphasis on component failure, is inappropriate for today's software-intensive, complex human-machine systems with distributed decision-making across both physical and organizational boundaries.

The basic premise of this paper is that to make significant progress in dealing with safety in complex systems, we need new models and conceptions of how accidents occur that more accurately and completely reflect the types of accidents we are experiencing today. Simply building more tools based on the current chain-of-events model will not result in significant gains. This paper presents one example of such a model, but others are possible.

The new model, called STAMP (Systems-Theoretic Accident Model and Processes), uses a systems-theoretic approach to understanding accident causation. Systems theory allows more complex relationships between events to be considered (e.g., feedback and other indirect relationships) and also provides a way to look more deeply at why the events occurred. Accident models based on systems theory consider accidents as arising from the interactions among system components and usually do not specify single causal variables or factors [9]. Whereas industrial (occupational) safety models focus on unsafe acts or conditions and reliability engineering emphasizes failure events and the direct relationships between these events, a systems approach to safety takes a broader view by focusing on what was wrong with the system's design or operations that allowed the accident to take place. The proximal events that precede an accident are simply symptoms of a lack of enforcement of safety in the design and operation of the system: to prevent accidents we need to go beyond the events to understand why those events occurred, i.e., the larger system and process producing the events.

The next section provides some basic background on system theory, followed by a description

of a systems-theoretic approach to safety. The basic concepts are illustrated using the loss of a Milstar satellite.

1.1 Safety as an Emergent System Property

Event chain models rest on traditional *analytic reduction*: Physical systems are decomposed into separate physical components so the parts can be examined separately, and behavior is decomposed into events over time. This decomposition assumes that such separation is feasible: that is, each component or subsystem operates independently and analysis results are not distorted when the components are considered separately. This assumption in turn implies (1) that the components or events are not subject to feedback loops and non-linear interactions and (2) that the behavior of the components is the same when examined alone as when they are playing their part in the whole. A third fundamental assumption is that the principles governing the assembly of the components into the whole are straightforward, that is, the interactions among the subsystems are simple enough that they can be considered separate from the behavior of the subsystems themselves [28].

These assumptions are reasonable for many properties and systems, but they start to fall apart in complex systems. Systems theory dates from the thirties and forties and was a response to limitations of the classic analysis techniques in coping with the increasingly complex systems being built [4]. Norbert Weiner applied the approach to control and communications engineering [29] while Ludwig von Bertalanffy developed similar ideas for biology [3]. It was Bertalanffy who suggested that the emerging ideas in various fields could be combined into a general theory of systems.

The systems approach focuses on systems taken as a whole, not on the parts examined separately. It assumes that some properties of systems can only be treated adequately in their entirety, taking into account all facets relating the social to the technical aspects [4]. These system properties derive from the relationships between the parts of systems: how the parts interact and fit together [1]. Thus the systems approach concentrates on the analysis and design of the system as a whole as distinct from the components or the parts. While components may be constructed in a modular fashion, the original analysis and decomposition must be performed top down.

The foundation of systems theory rests on two pairs of ideas: (1) *emergence* and *hierarchy* and (2) *communication* and *control* [4].

1.1.1 Emergence and Hierarchy

The first pair of basic system theory ideas are emergence and hierarchy. A general model 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. Emergent properties do not exist at lower levels; they are meaningless in the language appropriate to those levels. 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. Thus, the operation of the processes at the lower levels of the hierarchy result in a higher level of complexity—that of the whole apple itself—that has emergent properties, one of them being the apple's shape. The concept of emergence is the idea that at a given level of complexity, some properties characteristic of that level (emergent at that level) are irreducible.

Safety is an emergent property of systems. Determining whether a plant is acceptably safe is not possible by examining a single valve in the plant. In fact, statements about the "safety of

the valve" without information about the context in which that valve is used, are meaningless. Conclusions can be reached, however, about the reliability of the valve, where reliability is defined as "the ability of a system or component to perform its required functions under stated conditions for a specified period of time" [8], i.e., that the behavior of the valve will satisfy its specification over time and under given conditions. This is one of the basic distinctions between safety and reliability: Safety can only be determined by the relationship between the valve and the other plant components—that is, in the context of the whole. Therefore it is not possible to take a single system component, like a software module, in isolation and assess its safety. A component that is perfectly safe in one system may not be when used in another. Attempts to assign safety levels to software components in isolation from a particular use, as is currently the approach in some international safety standards, is misguided.

Event-based models of accidents, with their relatively simple cause-effect links, were created in an era of mechanical systems and then adapted for electro-mechanical systems. The use of software in engineered systems has removed many of the physical constraints that limit complexity and has allowed engineers to incorporate greatly increased complexity and coupling in systems containing large numbers of dynamically interacting components. In the simpler systems of the past, where all the interactions between components could be predicted and handled, component failure was the primary cause of accidents. In today's complex systems, made possible by the use of software, this is no longer the case. The same applies to security and other system properties: While some vulnerabilities may be related to a single component only, a more interesting class of vulnerability emerges in the interactions among multiple system components. Vulnerabilities of this type are *system* vulnerabilities and are much more difficult to locate and predict.

A second basic part of systems theory, hierarchy theory, deals with the fundamental differences between levels of a hierarchy. Its ultimate aim is to explain the relationships between different levels: what generates the levels, what separates them, and what links them. Emergent properties associated with a set of components at one level in a hierarchy are related to *constraints upon the degree of freedom* of those components. In a systems-theoretic view of safety, the emergent safety properties are controlled or enforced by a set of safety constraints related to the behavior of the system components. Safety constraints specify those relationships among system variables or components that constitute the non-hazardous or safe system states—for example, the power must never be on when the access door to the high-voltage power source is open; pilots in a combat zone must always be able to identify potential targets as hostile or friendly; the public health system must prevent the exposure of the public to contaminated water; and the spacecraft lander software must control the rate of descent of the spacecraft to the planet's surface. Accidents result from interactions among system components that violate these constraints—in other words, from a lack of appropriate constraints on system behavior.

1.1.2 Communication and Control

The second pair of basic systems theory ideas is communication and control. Regulatory or *control* action is the imposition of constraints upon the activity at one level of a hierarchy, which define the "laws of behavior" at that level yielding activity meaningful at a higher level. Hierarchies are characterized by control processes operating at the interfaces between levels. Checkland writes:

> Control is always associated with the imposition of constraints, and an account of a control process necessarily requires our taking into account at least two hierarchical

> levels. At a given level, it is often possible to describe the level by writing dynamical equations, on the assumption that one particle is representative of the collection and that the forces at other levels do not interfere. But any description of a control process entails an upper level imposing constraints upon the lower. The upper level is a source of an alternative (simpler) description of the lower level in terms of specific functions that are emergent as a result of the imposition of constraints [4, p.87].

Control in open systems (those that have inputs and outputs from their environment) implies the need for *communication*. Bertalanffy distinguished between *closed systems*, in which unchanging components settle into a state of equilibrium, and *open systems*, which can be thrown out of equilibrium by exchanges with their environment [3]. The notions of time lag, noise, and bandwidth play important roles in communication between hierarchical control levels.

In systems theory, open systems are viewed as interrelated components that are kept in a state of dynamic equilibrium by feedback loops of information and control. A system is not treated as a static design, but as a dynamic process that is continually adapting to achieve its ends and to react to changes in itself and its environment. To be safe, the original design must not only enforce appropriate constraints on behavior to ensure safe operation (the enforcement of the safety constraints), but it must continue to operate safely as changes and adaptations occur over time [30].

1.2 Systems-Theoretic Approaches to Safety

In response to the limitations of event-chain models, systems theory has been proposed as a way to understand accident causation (see, for example, [22, 11]). When using a systems-theoretic accident model, accidents are viewed as the result of flawed processes involving interactions among system components, including people, societal and organizational structures, engineering activities, and the physical system.

Rasmussen and Svedung [22, 23] have added some features of system theory into the classic event-chain model by adding hierarchical control levels—representing government, regulators and associations, company, management, and staff—above the event chain (Figure 1). Information flow is mapped at all levels and between levels. The model concentrates on the operation of the socio-technical system: information from the system design and analysis process (the left column in the figure) is treated as input to the operations process (the right column). At each level, they model the factors involved using event chains, with links to the chain at the level below. Unfortunately, retaining event chains and event decomposition at the hierarchical levels limits the benefits that can be obtained by taking a systems approach.

Leveson has defined another accident model, called STAMP (Systems-Theoretic Accident Modeling and Processes) based on systems theory [11]. In STAMP, accidents are conceived as resulting not from component failures, but from inadequate control or enforcement of safety-related constraints on the design, development, and operation of the system. In the Space Shuttle *Challenger* accident, for example, the O-rings did not adequately control propellant gas release by sealing a tiny gap in the field joint. In the Mars Polar Lander loss, the software did not adequately control the descent speed of the spacecraft—it misinterpreted noise from a Hall effect sensor as an indication the spacecraft had reached the surface of the planet.

Accidents such as these, involving engineering design errors, may in turn stem from inadequate control over the development process, i.e., risk is not adequately managed (through communication and feedback) in the design, implementation, and manufacturing processes. Control is also imposed by the management functions in an organization—the *Challenger* accident

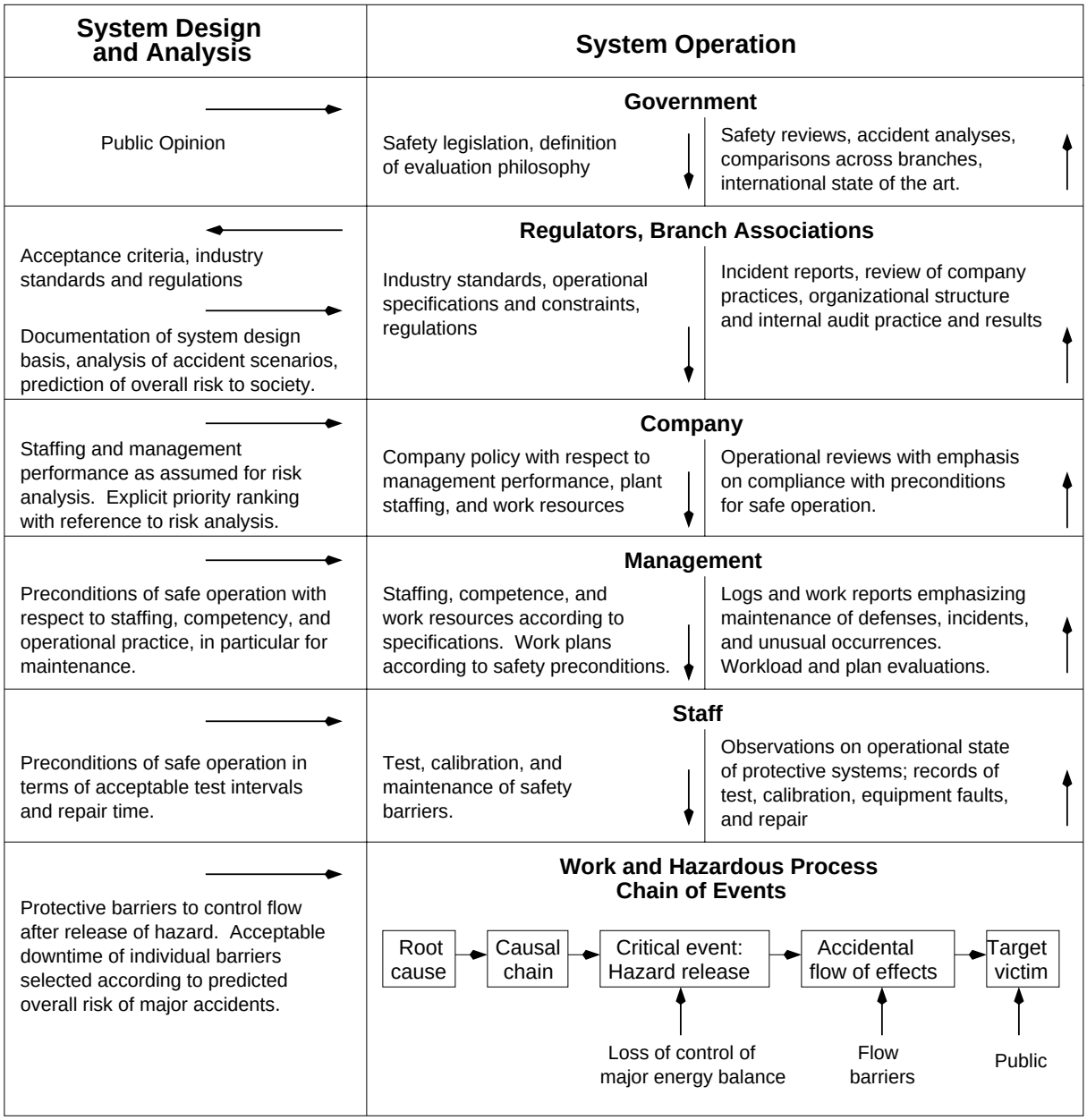

Figure 1: Rasmussen-Svedung Model

involved inadequate controls in the launch-decision process, for example—and by the social and political system within which the organization exists.

STAMP is constructed from three basic concepts: constraints, hierarchical levels of control, and process models. These concepts, in turn, give rise to a classification of control flaws that can lead to accidents. Each of these is described below.

1.2.1 Constraints

The most basic concept in STAMP is not an event, but a constraint. In systems theory or control 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 [4]. Safety-related constraints specify those relationships among system variables that constitute the nonhazardous or safe system states.

Instead of viewing accidents as the result of an initiating (root cause) event in a chain of events leading to a loss (which must somehow be broken in order to prevent them), accidents are viewed as resulting from interactions among components that violate the system safety constraints. The control processes that enforce these constraints must limit system behavior to the safe changes and adaptations implied by the constraints.

Note that accidents caused by basic component failures are included in this model, as well as those caused by interactions among components. Identifying the failure events themselves, however, does not provide enough information about why they occurred to prevent similar accidents in the future. 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 task requirements; unhandled environmental disturbances (e.g., electromagnetic inteference or EMI); inadequate maintenance, including preventive maintenance; physical degradation over time (wearout); etc. Control therefore need not be imposed by a physical "controller" but may be controlled through system design or manufacturing processes and procedures. Systems-theoretic accident models go beyond simply blaming component failure for accidents (and perhaps then adding redundancy to the design to handle them) and require that the reasons be identified for why those failures occur and lead to accidents.

1.2.2 Hierarchical Levels of Control

The second concept in STAMP (and a basic concept in systems theory) is hierachical levels of control. Figure 2 shows a generic hierarchical safety control model. Accidents result from inadequate enforcement of constraints on behavior (e.g., the physical system, engineering design, management, and regulatory behavior) at each level of the socio-technical system.

The model in Figure 2 has two basic hierarchical control structures—one for system development (on the left) and one for system operation (on the right)—with interactions between them. An aircraft manufacturer, for example, might only have system development under its immediate control, but safety involves both development and operational use of the aircraft, and neither can be accomplished successfully in isolation: Safety must be designed into the system, and safety during operation depends partly on the original design and partly on effective control over operations and the changes and adaptations in the system over time. Manufacturers must communicate to their customers the assumptions about the operational environment upon which the safety analysis was based, as well as information about safe operating procedures. The operational environment, in turn, provides feedback to the manufacturer about the perfor-

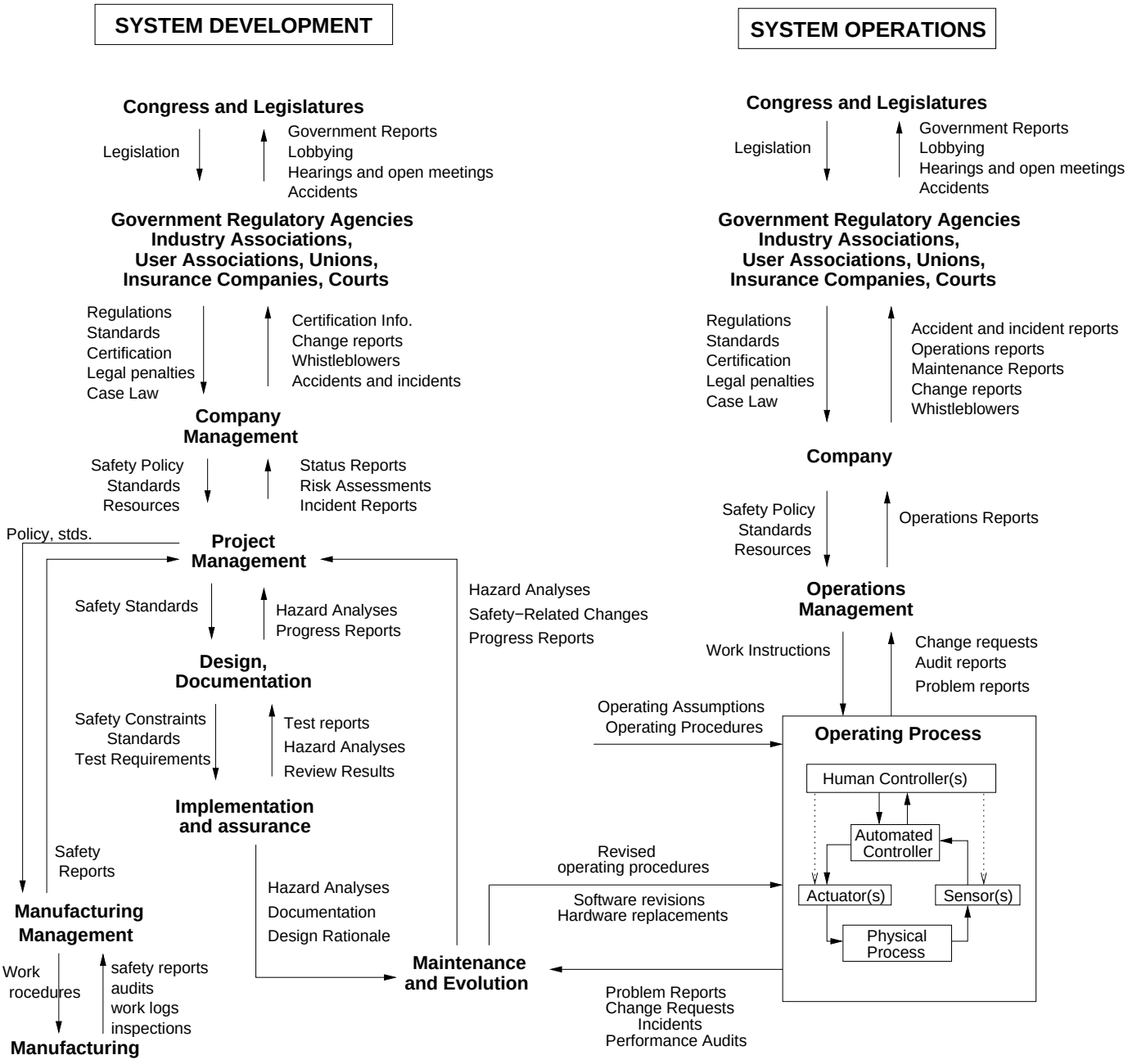

Figure 2: General Form of a Model of Socio-Technical Control.

mance of the system during operations. Although difficult to show without excessively cluttering the figure, interaction may occur at all levels between the development and operations control structures.

Between the hierarchical levels of each control structure, effective communication channels are needed, both a downward *reference* channel providing the information necessary to impose constraints on the level below and a *measuring* channel to provide feedback about how effectively the constraints were enforced. For example, company management in the development process structure may provide a safety policy, standards, and resources to project management and in return receive status reports, risk assessment, and incident reports as feedback about the status of the project with respect to the safety constraints. As described later, time lag, noise, and bandwidth must be considered when analyzing the performance of the communication channels.

The safety control structure often changes over time, which accounts for the observation 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 [25]: One event may trigger the loss, but if that event had not happened, another one would have. Union Carbide and the Indian government blamed the Bhopal MIC (methyl isocyanate) 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 [10]. Instead, degradation in the safety margin 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:

> The stage for an accidental course of events very likely is prepared through time by the normal efforts of many actors in their respective daily work context, responding to the standing request to be more productive and less costly. Ultimately, a quite normal variation in somebody's behavior can then release an accident. Had this 'root cause' been avoided by some additional safety measure, the accident would very likely be released by another cause at another point in time. In other words, an explanation of the accident in terms of events, acts, and errors is not very useful for design of improved systems [24].

Degradation of the safety-control structure over time may be related to *asynchronous evolution* [9], 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 system software did not [15]. 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 software procedure to perform gyro spin-down [17].

For an accident model to handle system adaptation over time, it 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 as it changes and adapts over time rather than considering individual events and human actions.

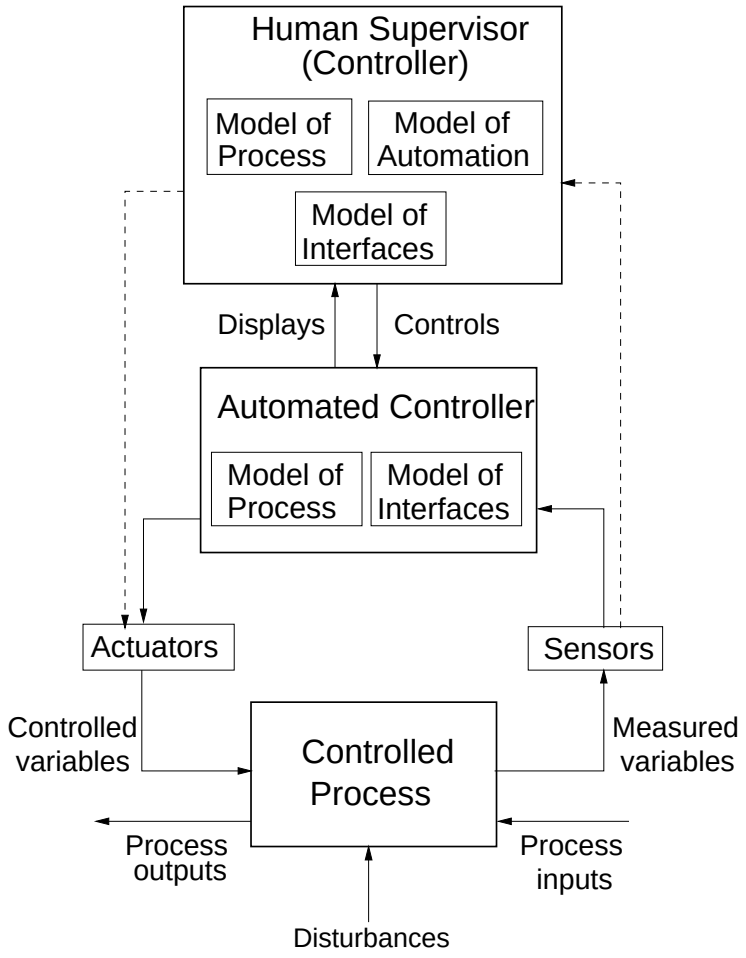

Figure 3: A standard hierarchical three-level control loop.

1.2.3 Process Models

Besides constraints and hierarchical levels of control, a third basic concept in STAMP is that of process models. Figure 3 shows a typical process-control loop with an automated controller supervised by a human controller. Any controller—human or automated—needs a model of the process being controlled to effectively control it. The model may contain only one or two state variables, such as the model required for a simple thermostat, which contains the current temperature and the desired setpoint, or it may be very complex, such as the model of the airspace required for air traffic control. Human controllers of automated systems must have an additional model of the automation as well as the controlled process, and both the human controller and the software need models of the interfaces between system components.

Whether the model is embedded in the control logic of an automated controller or in the mental model of a human controller, it must contain the same type of information: the required relationship among the system variables (the control laws), the current state (the current values of the system variables), and the ways the process can change state. This model is used to determine what control actions are needed, and it is updated through various forms of feedback. A model of the controlled process is required at all levels of the hierarchical control structure.

There may, of course, be multiple human and automated controllers in the control loop, and computers may play roles other than as a direct controller. For example, computers may act as automated decision aids that provide information to the human controller but do not directly issue control commands. If the computer provides decision aiding, then the software must still contain a model of the process because it is indirectly controlling the process.

Time is an important consideration; 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; the process may have delays in responding to manipulated variables; and the sensors may obtain values only a certain sampling intervals. Time lags restrict the speed and extent with which the effects of disturbances, both within the process itself and externally derived, can be reduced. They also impose extra requirements on the controller, for example, the need to infer delays that are not directly observable. Accidents can occur due to inadequate handling of these delays. Noise and bandwidth can similarly impact performance of the control loop.

1.2.4 A Classification of Control Flaws Leading to Accidents

In basic systems theory, to effect control over a system requires four conditions [2, 5]:

- **Goal Condition**: The controller must have a goal or goals, e.g., to maintain the setpoint or to maintain the safety constraints.
- **Action Condition**: The controller must be able to affect the state of the system in order to keep the process operating within predefined limits or safety constraints despite internal or external disturbances. Where there are multiple controllers and decision makers, the actions must be coordinated to achieve the goal condition. Uncoordinated actions are particularly likely to lead to accidents in the boundary areas between controlled processes or when multiple controllers have overlapping control responsibilities.
- **Model Condition**: The controller must be (or contain) a model of the system, as described above. Accidents in complex systems frequently result from inconsistencies between the model of the process used by the controllers (both human and software) and the actual process state; for example, the software thinks the plane is climbing when it is actually descending and as a result applies the wrong control law or the pilot thinks a friendly aircraft is hostile and shoots a missile at it.
- **Observability Condition**: The controller must be able to ascertain the state of the system from information about the process state provided by *feedback*. Feedback is used to update and maintain the process model used by the controller.

Using systems theory, accidents can be understood in terms of failure to adequately satisfy these four conditions:

1. Hazards and the safety constraints to prevent them are not identified and provided to the controllers (goal condition);
2. The controllers are not able to effectively maintain the safety constraints or they do not make appropriate or effective control actions for some reason, perhaps because of inadequate coordination among multiple controllers (action condition);

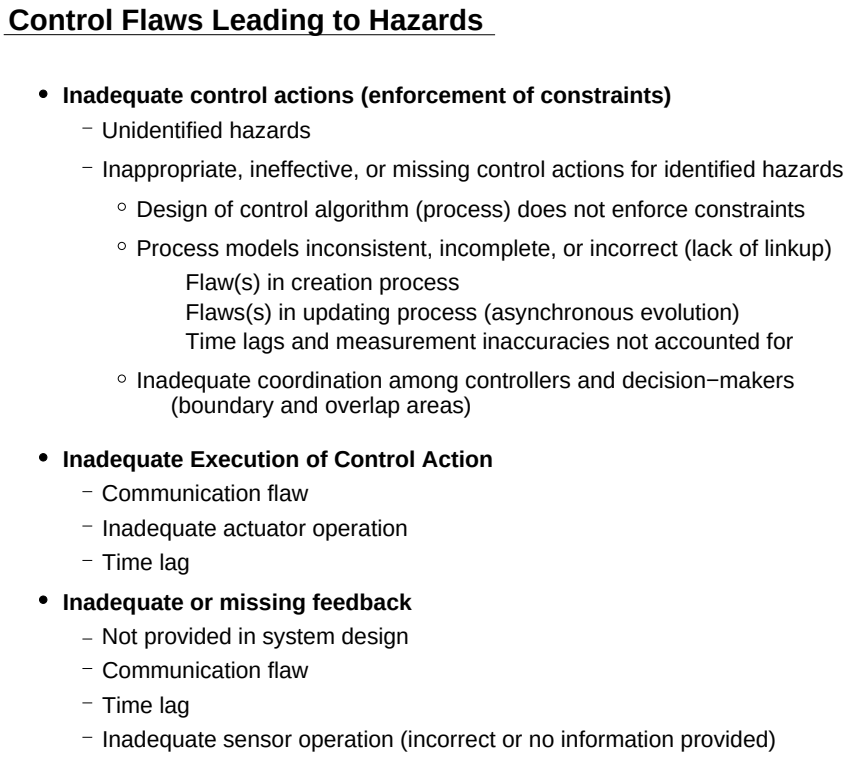

Figure 4: A classification of control flaws leading to accidents.

3. The process models used by the software or by human controllers (usually called mental models in the case of humans) become inconsistent with the process and with each other (model condition); and

4. The controller is unable to ascertain the state of the system and update the process models because feedback is missing or inadequate (observability condition).

When using a systems-theoretic accident model such as STAMP, the control flaws identified above are mapped to the components of the control loop and used in understanding and preventing accidents. Figure 4 shows a categorization of control flaws that can lead to the violation of the four conditions above. This categorization can be used in the creation of new hazard and accident analysis techniques (see, for example, [14, 6]).

2 Using a Systems-Theoretic Accident Model

The rest of this paper contains an extensive example that uses STAMP to understand the reasons for a software-related accident. On April 30, 1999, at 12:30 EDT, a Titan IV B-32 booster equipped with a Centaur TC-14 upper stage was launched from Cape Canaveral. The mission was to place a Milstar-3 satellite into geosynchronous orbit. Milstar is a joint services satellite communications system that provides secure, jam resistant, worldwide communications to meet wartime requirements. It was the most advanced military communications satellite system to that date. The first Milstar satellite was launched February 7, 1994 and the second was launched November 5, 1995. This mission was to be the third launch.

As a result of some anomalous events, the Milstar satellite was placed in an incorrect and unusable low elliptical final orbit, as opposed to the intended geosynchronous orbit. This accident is believed to be one of the most costly unmanned losses in the history of Cape Canaveral launch operations. The Milstar satellite cost about $800 million and the launcher an additional $433 million.

To their credit, the accident investigation board went beyond the usual chain-of-events model and instead interpreted the accident in terms of a complex and flawed process [18]:

> Failure of the Titan IV B-32 mission is due to a failed software development, testing, and quality assurance process for the Centaur upper stage. That failed process did not detect and correct a human error in the manual entry of the I1(25) roll rate filter constant entered in the Inertial Measurement System flight software file. The value should have been entered as -1.992476, but was entered as -0.1992476. Evidence of the incorrect I1(25) constant appeared during launch processing and the launch countdown, but its impact was not sufficiently recognized or understood and, consequently, not corrected before launch. The incorrect roll rate filter constant zeroed any roll rate data, resulting in the loss of roll axis control, which then caused loss of yaw and pitch control. The loss of attitude control caused excessive firings of the Reaction Control system and subsequent hydrazine depletion. Erratic vehicle flight during the Centaur main engine burns caused the Centaur to achieve an orbit apogee and perigee much lower than desired, which resulted in the Milstar separating in a useless low final orbit [18, p. 2].

Fully understanding this accident requires understanding why the error in the roll rate filter constant was introduced in the load tape, why it was not found during the load tape production process and internal review processes, why it was not found during the extensive independent verification and validation effort applied to this software, and why it was not detected during operations at the launch site. In other words, why the safety control structure was ineffective in each of these instances.

Figure 5 shows the hierarchical control model of the accident, or at least those parts that can be gleaned from the official accident report[3]. Lockheed Martin Astronautics (LMA) was the prime contractor for the mission. The Air Force Space and Missile Systems Center Launch Directorate (SMC) was responsible for insight and administration of the LMA contract. Besides LMA and SMC, the Defense Contract Management Command (DCMC) played an oversight role, but the report is not clear about what exactly this role was beyond a general statement about responsibility for contract management, software surveillance, and overseeing the development process.

LMA designed and developed the flight control software, while Honeywell was responsible for the IMS software. This separation of control, combined with poor coordination, accounts for some of the problems that occurred. Analex was the independent verification and validation (IV&V) contractor, while Aerospace Corporation provided independent monitoring and evaluation. Ground launch operations at Cape Canaveral Air Station (CCAS) were managed by the Third Space Launch Squadron (3SLS).

Starting from the physical process and working up the levels of control, an analysis based on a systems-theoretic accident model examines each level for the flaws in the process at that level that provided inadequate control of safety in the process level below. The process flaws

[3]Some details of the control structure may be incorrect because they were not detailed in the report, but the structure is close enough for the purpose of this paper.

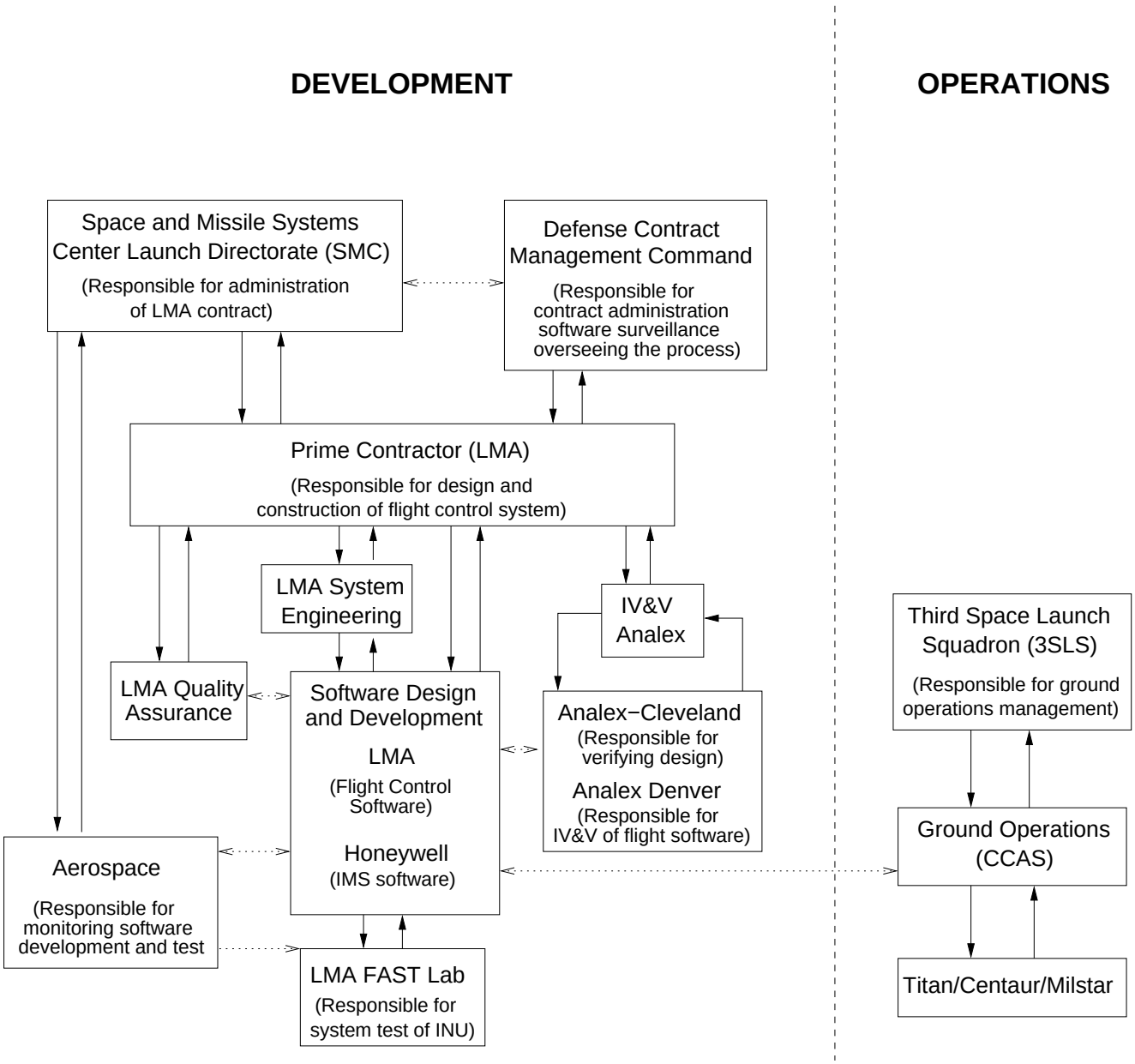

Figure 5: Hierarchical Control Structure

at each level are then examined and explained in terms of a potential mismatch between the controller's model of the process and the actual state of the process, incorrect design of the control algorithm, lack of coordination among the control activities, deficiencies in the reference channel, and deficiencies in the feedback or monitoring channel. When human decision-making is involved, the analysis results must also include information about the context in which the decision(s) was made and the information available and not available at the time the decision(s) was made. A detailed example follows.

2.1 The Physical Process (Titan/Centaur/Milstar)

Components of the Physical Process: The Lockheed Martin Astronautics (LMA) Titan IV B is a heavy-lift space launch vehicle used to carry government payloads such as Defense Support Program, Milstar, and National Reconnaissance Office satellites into space. It can carry up to 47,800 pounds into low-earth orbit and up to 12,700 pounds into a geosynchronous orbit. The vehicle can be launched with no upper stage or with one of two optional upper stages, providing greater and varied capability.

The LMA Centaur is a cryogenic, high-energy upper stage. It carries its own guidance, navigation, and control system, which measures the Centaur's position and velocity on a continuing basis throughout flight. It also determines the desired orientation of the vehicle in terms of pitch, yaw, and roll axis vectors. It then issues commands to the required control components to orient the vehicle in the proper attitude and position, using the main engine or the Reaction Control System (RCS) engines (Figure 6). The main engines are used to control thrust and velocity. The RCS provides thrust for vehicle pitch, yaw, and roll control, for post-injection separation and orientation maneuvers, and for propellant settling prior to engine restart.

System Hazards Involved: (1) The satellite does not reach a useful geosynchronous orbit; (2) the satellite is damaged during orbit insertion maneuvers and cannot provide its intended function.

Description of Process Controller (INU): The Inertial Navigation Unit (INU) has two parts (Figure 6): (1) the Guidance, Navigation, and Control System (the Flight Control Software or FCS) and (2) an Inertial Measurement System (IMS). The Flight Control Software computes the desired orientation of the vehicle in terms of the pitch, yaw, and roll axis vectors and issues commands to the main engines and the reaction control system to control vehicle orientation and thrust. To accomplish this goal, the FCS uses position and velocity information provided by the IMS. The component of the IMS involved in the loss is a roll rate filter, which is designed to prevent the Centaur from responding to the effects of Milstar fuel sloshing and thus inducing roll rate errors.

Safety Constraint on FCS: The FCS must provide the attitude control, separation, and orientation maneuvering commands to the main engines and the RCS system necessary to attain geosynchronous orbit.

Safety Constraint on IMS: The position and velocity values provided to the FCS must not be capable of leading to a hazardous control action. The roll rate filter must prevent the Centaur from responding to the effects of fuel sloshing and inducing roll rate errors.

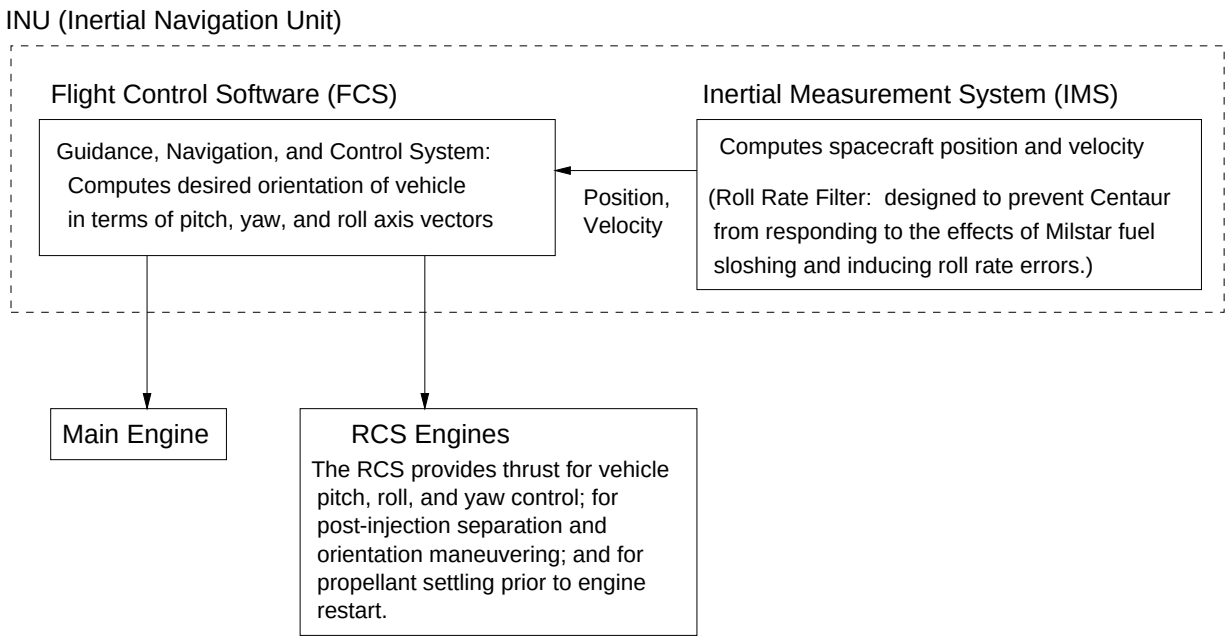

Figure 6: Technical Process Control Structure for INU

2.2 Description of the Proximal Events Leading to the Loss

There were three planned burns during the Centaur flight. The first burn was intended to put the Centaur into a parking orbit. The second would move the Centaur into an elliptical transfer orbit that was to carry the Centaur and the satellite to geosynchronous orbit. The third and final burn would circularize the Centaur in its intended geosynchronous orbit. A coast phase was planned between each burn. During the coast phase, the Centaur was to progress under its own momentum to the proper point in the orbit for the next burn. The Centaur would also exercise a roll sequence and an attitude control maneuver during the coast periods to provide passive thermal control and to settle the main engine propellants in the bottom of the tanks.

First Burn: The first burn was intended to put the Centaur into a parking orbit. The Inertial Measurement System (IMS) transmitted a zero or near zero roll rate to the Flight Control Software (FCS), however, due to the use of an incorrect roll rate filter constant. With no roll rate feedback, the FCS provided inappropriate control commands that caused the Centaur to become unstable about the roll axis and not to roll to the desired first burn orientation. The Centaur began to roll back and forth, eventually creating sloshing of the vehicle liquid fuel in the tanks, which created unpredictable forces on the vehicle and adversely affected flow of fuel to the engines. By the end of the first burn (approximately 11 minutes and 35 seconds after liftoff), the roll oscillation began to affect the pitch and yaw rates of the vehicle as well. The FCS predicted an incorrect time for main engine shutdown due to the effect on the acceleration of the vehicle's tumbling and fuel sloshing. The incorrect shutdown in turn resulted in the Centaur not achieving its intended velocity during the first burn, and the vehicle was placed in an unintended park orbit.

First Coast Phase: During the coast phases, the Centaur was to progress under its own momentum to the proper point in the orbit for the next burn. During this coasting period, the FCS was supposed to command a roll sequence and an attitude control maneuver to provide passive thermal control and to settle the main engine propellants in the bottom of the tanks.

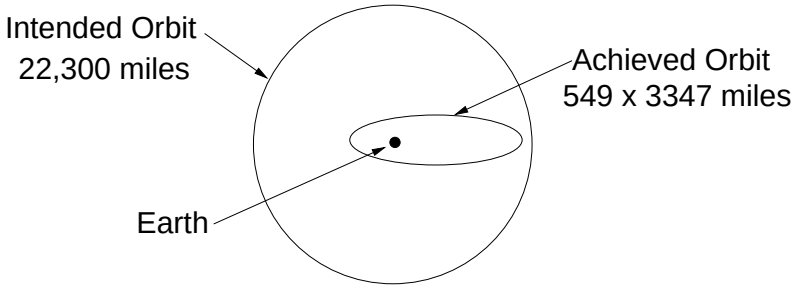

Figure 7: Achieved Orbit vs. Intended Orbit

Because of the roll instability and transients created by the engine shutdown, the Centaur entered this first coast phase tumbling. The FCS directed the RCS to stabilize the vehicle. Late in the park orbit, the Centaur was finally stablized about the pitch and yaw axes, although it continued to oscillate about the roll axis. In stabilizing the vehicle, however, the RCS expended almost 85 percent of the RCS system propellant (hydrazine).

Second Burn: The FCS successfully commanded the vehicle into the proper attitude for the second burn, which was to put the Centaur and the satellite into an elliptical transfer orbit that would carry them to geosynchronous orbit. The FCS ignited the main engines at approximately one hour, six minutes, and twenty-eight seconds after liftoff. Soon after entering the second burn phase, however, inadequate FCS control commands caused the vehicle to again become unstable about the roll axis and begin a diverging roll oscillation.

Because the second burn is longer than the first, the excess roll commands from the FCS eventually saturated the pitch and yaw channels. At approximately two minutes into the second burn, pitch and yaw control was lost (as well as roll), causing the vehicle to tumble for the remainder of the burn. Due to its uncontrolled tumbling during the burn, the vehicle did not achieve the planned acceleration for transfer orbit.

Second Coast Phase (transfer orbit): The RCS attempted to stabilize the vehicle but it continued to tumble. The RCS depleted its remaining propellant approximately twelve minutes after the FCS shut down the second burn.

Third Burn: The goal of the third burn was to circularize the Centaur in its intended geosynchronous orbit. The FCS started the third burn at two hours, thirty-four minutes, and fifteen seconds after liftoff. It was started earlier and was shorter than had been planned. The vehicle tumbled throughout the third burn, but without the RCS there was no way to control it. Space vehicle separation was commanded at approximately two hours after the third burn began, resulting in the Milstar being placed in a useless low elliptical orbit, as opposed to the desired geosynchronous orbit (Figure 7).

Post Separation: The Mission Director ordered early turn-on of the satellite in an attempt to save it, but the ground controllers were unable to contact the satellite for approximately three hours. Six hours and fourteen minutes after liftoff, control was acquired and various survival and emergency actions were taken. The satellite had been damaged from the uncontrolled vehicle pitch, yaw, and roll movements, however, and there were no possible actions the ground controllers could have taken in response to the anomalous events that would have saved the mission.

The mission was officially declared a failure on May 4, 1999, but personnel from LMA and the Air Force controlled the satellite for six additional days in order to place the satellite in a non-interfering orbit with minimum risk to operational satellites. It appears the satellite

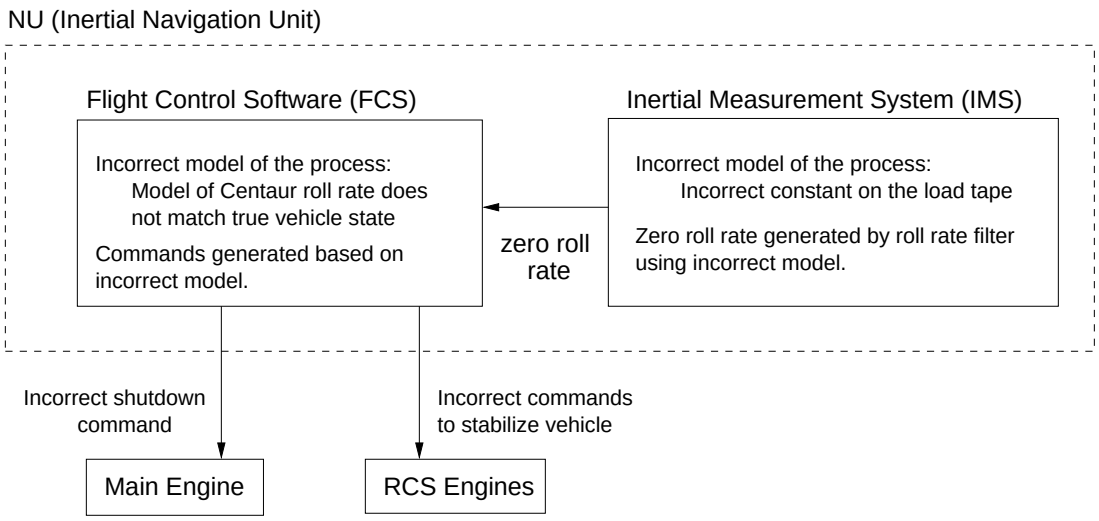

Figure 8: Control Flaws at the Physical Process and Software Controller Levels

performed as designed, despite the anomalous conditions. It was shut down by ground control on May 10, 1999.

2.3 Physical Process and Automated Controller Failures and Dysfunctional Interactions

Figure 8 shows the automated controller flaws leading to the accident. The Inertial Measurement System software used an incorrect model of the process (an incorrect roll rate filter constant in the IMS software file) that led to a dysfunctional interaction with the flight control software. However, the algorithm operated as designed (i.e., it did not fail).

The Flight Control Software operated correctly (i.e., according to its requirements). However, it received incorrect input from the IMS, leading to an incorrect internal FCS software model of the process—the roll rate was thought to be zero or near zero when it was not. Thus there was a mismatch between the FCS internal model of the process state and the real process state. This mismatch led to the RCS issuing incorrect control commands to the main engine (to shut down early) and to the RCS engines. Using STAMP terminology, the loss resulted from a dysfunctional interaction between the FCS and the IMS. Neither failed—they operated correctly with respect to the instructions (including constants) and data provided.

The accident report does not explore whether the FCS software could have included sanity checks on the roll rate or vehicle behavior to detect that incorrect roll rates were being provided by the IMS or checks to determine whether inputs to the FCS software were potentially destabilizing. Even if the FCS did detect it was getting anomalous roll rates, there may not have been any recovery or fail-safe behavior that could have been designed into the system. Without more information about the Centaur control requirements and design, it is not possible to speculate about whether the Inertial Navigation Unit software (the IMS and FCS) might have been designed to be fault tolerant with respect to filter constant errors.

This level of explanation of the flaws in the process (the vehicle and its flight behavior) as well as its immediate controller provides a description of the "symptom," but does not provide enough information about the factors involved to prevent reoccurrences. Simply fixing that particular flight tape is not enough. We need to look at the higher levels of the control structure

for that. Figures 9 and 10 summarize the information in the rest of this paper.

2.4 Launch Site Operations

The function of launch site operations is to monitor launch pad behavior and tests and to detect any critical anomalies prior to flight. Why was the roll rate error not detected during launch operations?

Safety Constraint Violated: Critical variables (including those in software) must be monitored and errors detected before launch. Potentially hazardous anomalies detected at the launch site must be formally logged and thoroughly investigated and handled.

Context: Management had greatly reduced the number of engineers working launch operations, and those remaining were provided with few guidelines as to how they should perform their job. The accident report says that their tasks were not defined by their management so they used their best engineering judgment to determine which tasks they should perform, which variables they should monitor, and how closely to analyze the data associated with each of their monitoring tasks.

Safety Controls: The controls are not described well in the report. From what is included, it does not appear that controls were implemented to monitor or detect software errors at the launch site although a large number of vehicle variables were monitored.

Roles and Responsibilities: The report is also not explicit about the roles and responsibilities of those involved. LMA had launch personnel at CCAS, including Product Integrity Engineers (PIEs). 3SLS had launch personnel to control the launch process as well as software to check process variables and to assist the operators in evaluating observed data.

Failures, Dysfunctional Interactions, Flawed Decisions, and Inadequate Control Actions: Despite clear indications of a problem with the roll rate information being produced by the IMS, it was not detected by some launch personnel who should have and detected but mishandled by others. Specifically:

1. One week before launch, LMA personnel at CCAS observed much lower roll rate filter values than they expected. When they could not explain the differences at their level, they raised their concerns to Denver LMA Guidance Product Integrity Engineers (PIEs), who were now at CCAS. The on-site PIEs could not explain the differences either, so they directed the CCAS personnel to call the control dynamics (CD) design engineers in Denver. On Friday, April 23, the LMA Guidance Engineer telephoned the LMA CD lead. The CD lead was not in his office so the Guidance Engineer left a voice mail stating she noticed a significant change in roll rate when the latest filter rate coefficients were entered. She requested a return call to her or to her supervisor. The Guidance Engineer also left an email for her supervisor at CCAS explaining the situation. Her supervisor was on vacation and was due back at the office Monday morning April 26, when the Guidance Engineer was scheduled to work the second shift. The CD lead and the CD engineer who originally specified the filter values listened to the voice mail from the Guidance Engineer. They called her supervisor at CCAS who had just returned from vacation. He was initially unable to find the email during their conversation. He said he would call back, so the

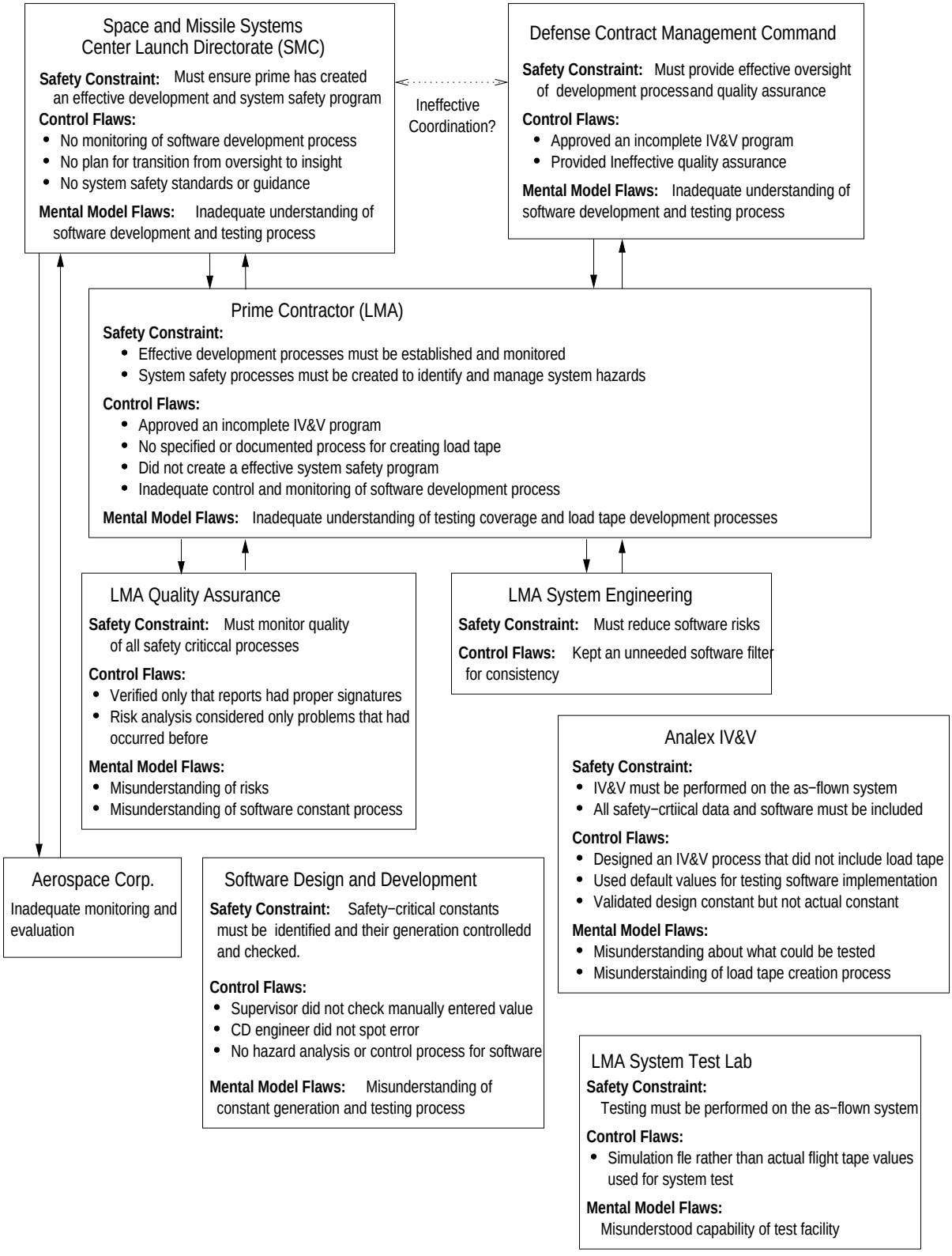

Figure 9: STAMP model of Development Process

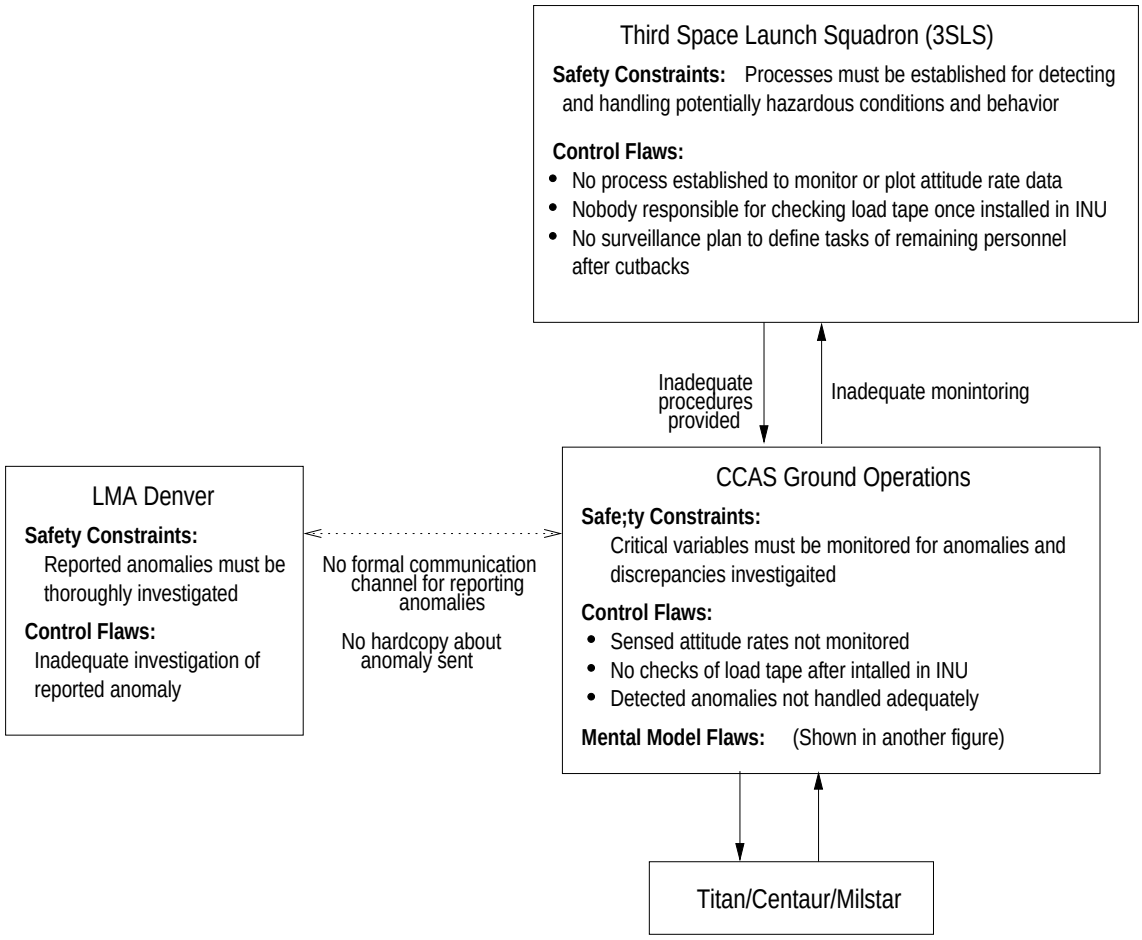

Figure 10: STAMP model of Launch Operations Process

CD engineer left the CD lead's office. The CD lead subsequently talked to the Guidance Engineer's supervisor after he found and read the email. The CD lead told the supervisor at CCAS that the filter values had changed in the flight tape originally loaded on April 14, 1999, and the roll rate output should also be expected to change. Both parties believed the difference in roll rates observed were attributable to expected changes with the delivery of the flight tape.

2. On the day of the launch, a 3SLS INU Product Integrity Engineer (PIE) at CCAS noticed the low roll rates and performed a rate check to see if the gyros were operating properly. Unfortunately, the programmed rate check used a default set of I1 constants to filter the measured rate and consequently reported that the gyros were sensing the earth rate correctly. If the sensed attitude rates had been monitored at that time or if they had been summed and plotted to ensure they were properly sensing the earth's gravitational rate, the roll rate problem could have been identified.

3. A 3SLS engineer also saw the roll rate data at the time of tower rollback, but was not able to identify the problem with the low roll rate. He had no documented requirement or procedures to review the data and no reference to compare to the roll rate actually being produced.

The communication channel between LMA Denver and the LMA engineers at CCAS was clearly flawed. There is no information about any established reporting channel from the LMA CCAS or LMA Denver engineers to a safety organization or up the management chain. No "alarm" system adequate to detect the problem or that it was not being adequately handled seems to have existed. The report says there was confusion and uncertainty from the time the roll rate anomaly was first raised by the CCAS LMA engineer in email and voice mail until it was "resolved" as to how it should be be reported, analyzed, documented, and tracked since it was a "concern" and not a "deviation." There is no explanation of these terms nor any description of a formal problem reporting and handling system in the accident report.

Inadequate Control Algorithm: The accident report says that at this point in the prelaunch process, there was no process to monitor or plot attitude rate data, that is, to perform a check to see if the attitude filters were properly sensing the earth's rotation rate. Nobody was responsible for checking the load tape constants once the tape was installed in the INU at the launch site. Therefore, nobody was able to question the anomalous rate data recorded or correlate it to the low roll rates observed about a week prior to launch and on the day of launch. In addition, the LMA engineers at Denver never asked to see a hard copy of the actual data observed at CCAS, nor did they talk to the guidance engineer or Data Station Monitor at CCAS who questioned the low filter rates. They simply explained it away as attributable to expected changes associated with the delivery of the flight tape.

Process Model Flaws: Five models are involved here (see Figure 11):

1. Ground rate check software: The software used to do a rate check on the day of launch used default constants instead of the actual load tape. Thus there was a mismatch between the model used in the rate checking software and the model used by the IMS software.

2. Ground crew models of the development process: Although the report does not delve into this factor, it is very possible that complacency may have been involved and that the model

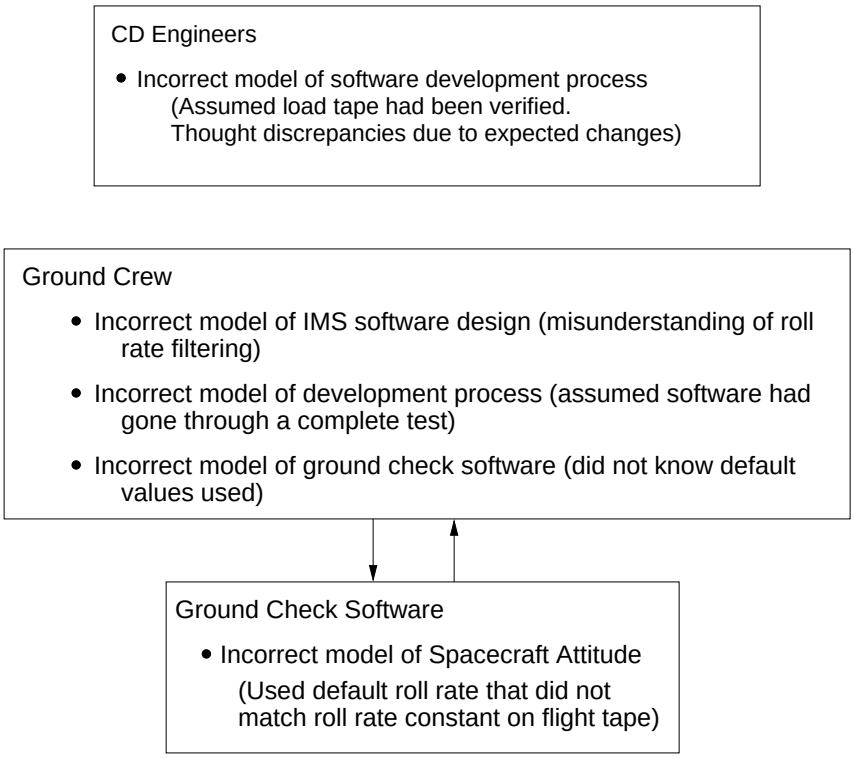

Figure 11: The Flawed Process Models used by the Ground Personnel and Software

of the thoroughness of the internal quality assurance and external IV&V development process in the minds of the ground operations personnel as well as the LMA guidance engineers who were informed of the observed anomalies right before launch did not match the real development process. There seemed to be no checking of the correctness of the software after the standard testing during development. Hardware failures are usually checked up to launch time, but often testing is assumed to have removed all software errors and therefore further checks are not needed.

3. Ground crew models of the IMS software design: The ground launch crew had an inadequate understanding of how the roll rate filters worked. No one other than the control dynamics engineers who designed the I1 roll rate constants understood their use or the impact of filtering the roll rate to zero. So when discrepancies were found before launch, nobody at the launch site understood the I1 roll rate filter design well enough to detect the error.

4. Ground crew models of the rate check software: Apparently, the ground crew was unaware that the checking software used default values for the filter constants.

5. CD engineers' model of the flight tape change: The control dynamics lead engineer at the launch site and her supervisor at LMA Denver thought that the roll rate anomalies were due to known changes in the flight tape. Neither went back to the engineers themselves to check this conclusion with those most expert in the details of the Centaur control dynamics.

Coordination: Despite several different groups being active at the launch site, nobody had been assigned responsibility for monitoring the software behavior after it was loaded into the

INU. The accident report does not mention coordination problems, although it does say there was a lack of understanding of each other's responsibilities between the LMA launch personnel (at CCAS) and the development personnel at LMA Denver and that this led to the concerns of the LMA personnel at CCAS not being adequately addressed.

A more general question that might have been investigated was whether the failure to act properly after detecting the roll rate problem involved a lack of coordination and communication problems between LMA engineers at CCAS and 3SLS personnel. Why did several people notice the problem with the roll rate but do nothing and why were the anomalies they noticed not effectively communicated to those who could do something about it? Several types of coordination problems might have existed. For example, there might have been an overlap problem, with each person who saw the problem assuming that someone else was handling it or the problem might have occurred at the boundary between several people's responsibilities.

Feedback: There was a missing or inadequate feedback channel from the launch personnel to the development organization.

Tests right before launch detected the zero roll rate, but there was no formal communication channel established for getting that information to those who could understand it. Instead voice mail and email were used. The report is not clear, but either there was no formal anomaly reporting and tracking system or it was not known or used by the process participants.[4]

The LMA (Denver) engineers requested no hardcopy information about the reported anomaly and did not speak directly with the Guidance engineer or Data Station Monitor at CCAS.

2.5 Air Force Launch Operations Management: Third Space Launch Squadron (3SLS)

Safety Constraint: Processes must be established for detecting and handling potentially hazardous conditions and behavior detected during launch preparations.

Context: 3SLS management was transitioning from an *oversight* role to an *insight* one without a clear definition of what such a transition might mean or require.

Control Algorithm Flaws: After the ground launch personnel cutbacks, 3SLS management did not create a master surveillance plan to define the tasks of the remaining personnel (the formal insight plan was still in draft). In particular, there were no formal processes established to check the validity of the I1 filter constants or to monitor attitude rates once the flight tape was loaded into the INU at Cape Canaveral Air Station (CCAS) prior to launch. 3SLS launch personnel were provided with no documented requirement nor procedures to review the data and no references with which to compare the observed data in order to detect anomalies.

Process Model: It is possible that misunderstandings (an incorrect model) about the thoroughness of the development process led to a failure to provide requirements and processes for performing software checks at the launch site. Complacency may also have been involved, i.e., the common assumption that software does not fail and that software testing is exhaustive and therefore additional software checking was not needed. However, this is speculation as the

[4]Several recent aerospace accidents have involved the bypassing of formal anomaly reporting channels and the substitution of informal email and other communication—with similar results [12].

report does not explain why management did not provide documented requirements and procedures to review the launch data nor ensure the availability of references for comparison so that discrepancies could be discovered.

Coordination: The lack of oversight led to a process that did not assign anyone the responsibility for some specific launch site tasks.

Feedback or Monitoring Channel: Apparently, launch operations management had no "insight" plan in place to monitor the performance of the launch operations process. There is no information included in the accident report about the process to monitor the performance of the launch operations process or what type of feedback was used (if any) to provide insight into the process.

2.6 Software/System Development of the Centaur Flight Control System

Too often, accident investigators stop at this point after identifying operational or sometimes maintenance errors that, if they had not occurred, might have prevented the loss [13]. Occasionally operations management is faulted. System design errors are much less likely to be identified. As an example, in the crash of an American Airlines DC-10 at Chicago's O'Hare Airport in 1979, the U.S. National Transportation Safety Board blamed only a "maintenance-induced crack" and not also a design error that allowed the slats to retract if the wing was punctured. Because of this omission, McDonnell Douglas was not required the change the design, leading to future accidents related to the same design error [19]. More recently, all of the Airbus A-320 accidents have been blamed on pilot error. Almost all, however, could equally be attributed to system and software design flaws. After the accidents, the software and sometimes the interface was modified to reduce the likelihood of the system design contributing to a human error.

Operator errors provide a convenient place to stop in the backward chain of events from the loss event. To their credit, the accident investigation board in the Titan/Milstar loss kept digging. To understand why an erroneous flight tape was created in the first place (and to learn how to prevent a similar occurrence in the future), the software and system development process associated with generating the tape needs to be examined.

Process Description: The INU consists of two major software components developed by different companies: LMA developed the Flight Control System software and was responsible for overall INU testing while Honeywell developed the IMS and was partially responsible for its software development and testing. The I1 constants are processed by the Honeywell IMS, but were designed and tested by LMA.

Safety Constraint Violated: Safety-critical constants must be identified and their generation controlled and checked.

Dysfunctional Interactions, Flawed Decisions, and Inadequate Control Actions: A Software Constants and Code Words Memo was generated by the LMA Control Dynamics (CD) group and sent to the LMA Centaur Flight Software (FS) group on December 23, 1997. It provided the intended and correct values for the first I1 constants in hardcopy form. The memo also allocated space for 10 additional constants to be provided by the LMA Avionics group at a later time and specified a path and file name for an electronic version of the first 30 constants.

The memo did not specify or direct the use of either the hardcopy or the electronic version for creating the constants database.

In early February, 1999, the LMA Centaur FS group responsible for accumulating all the software and constants for the flight load tape was given discretion in choosing a baseline data file. The flight software engineer who created the database dealt with over 700 flight constants generated by multiple sources, in differing formats, and at varying time (some with multiple iterations) all of which had to be merged into a single database. Some constant values came from electronic files that could be merged into the database, while others came from paper memos manually input into the database.

When the FS engineer tried to access the electronic file specified in the Software Constants and Code Words Memo, he found the file no longer existed at the specified location on the electronic file folder because it was now over a year after the file had been originally generated. The FS engineer selected a different file as a baseline that only required him to change five I1 values for the digital roll rate filter (an algorithm with five constants). The filter was designed to prevent the Centaur from responding to the effects of Milstar fuel sloshing and inducing roll rate errors at 4 radians/second. During manual entry of those five I1 roll rate filter values, the LMA FS engineer incorrectly entered or missed the exponent for the I1(25) constant. The correct value of the I1(25) filter constant was -1.992476. The exponent should have been a one but instead was entered as a zero, making the entered constant one tenth of the intended value or -0.1992476. The flight software engineer's immediate supervisor did not check the manually entered values.

The only person who checked the manually input I1 filter rate values, besides the flight software engineer who actually input the data, was an LMA Control Dynamics engineer. The FS engineer who developed the Flight Load tape notified the CD engineer responsible for design of the first thirty I1 constants that the tape was completed and the printout of the constants was ready for inspection. The CD engineer went to the FS offices and looked at the hardcopy listing to perform the check and sign off the I1 constants. The manual and visual check consisted of comparing a list of I1 constants from Appendix C of the Software Constants and Code Words Memo to the paper printout from the Flight Load tape. The formats of the floating-point numbers (the decimal and exponent formats) were different on each of these paper documents for the three values cross-checked for each I1 constant. The CD engineer did not spot the exponent error for I1(25) and signed off that the I1 constants on the Flight Load tape were correct. He did not know that the design values had been inserted manually into the database used to build the flight tapes (remember, the values had been stored electronically but the original database no longer existed) and that they were never formally tested in any simulation prior to launch.

The CD engineer's immediate supervisor, the lead for the CD section, did not review the Signoff Report nor catch the error. Once the incorrect filter constant went undetected in the Signoff Report, there were no other formal checks in the process to ensure the I1 filter rate values used in flight matched the designed filter.

Control Algorithm Flaws:

- A process input was missing (the electronic file specified in the Software Constants and Code Words memo), so an engineer regenerated it, making a mistake in doing so.

- Inadequate control was exercised over the constants process. No specified or documented software process existed for electronically merging all the inputs into a single file. There

was also no formal, documented process to check or verify the work of the flight software engineer in creating the file. Procedures for creating and updating the database were left up to the flight software engineer's discretion.

- Once the incorrect filter constant went undetected in the Signoff Report, there were no other formal checks in the process to ensure the I1 filter rate values used in flight matched the designed filter.

- The hazard analysis process was inadequate, and no control was exercised over the potential hazard of manually entering incorrect constants, a very common human error. If system safety engineers had identified the constants as critical, then a process would have existed for monitoring the generation of these critical variables. In fact, neither the existence of a system safety program nor any form of hazard analysis are mentioned in the accident report. If such a program had existed, one would think it would be mentioned.

  The report does say that Quality Assurance engineers performed a risk analysis, but they considered only those problems that had happened before:

  > Their risk analysis was not based on determining steps critical to mission success, but on how often problems previously surfaced in particular areas on past launches. They determined software constant generation was low risk because there had not been previous problems in that area. They only verified that the signoff report containing the constants had all the proper signatures[18].

Process Model Flaws: The accident report suggests that many of the various partners were confused about what the other groups were doing. The LMA software personnel who were responsible for creating the database (from which the flight tapes are generated) were not aware that IV&V testing did not use the as-flown (manually input) I1 filter constants in their verification and validation process. The LMA Control Dynamics engineer who designed the I1 rate filter also did not know that the design values were manually input into the database used to build the flight tapes and that the values were never formally tested in any simulation prior to launch.

While the failure of the LMA CD engineer who designed the I1 rate filter to find the error during his visual check was clearly related to the difficulty of checking long lists of differently formatted numbers, it also may have been partly due to less care being taken in the process due to an incorrect mental model, i.e., (1) he did not know the values were manually entered into the database (and were not from the electronic file he had created), (2) he did not know the load tape was never formally tested in any simulation prior to launch, and (3) he was unaware the load tape constants were not used in the IV&V process.

Coordination: The fragmentation/stovepiping in the flight software development process, coupled with the lack of comprehensive and defined system and safety engineering processes, resulted in poor and inadequate communication and coordination among the many partners and subprocesses.

Because the IMS software was developed by Honeywell, most everyone (LMA control dynamics engineers, flight software engineers, product integrity engineers, SQA, IV&V, and DCMC personnel) focused on the FCS and had little knowledge of the IMS software.

2.7 Quality Assurance (QA)

Safety Constraint: QA must monitor the quality of all safety-critical processes.

Process Flaw: The internal LMA quality assurance processes did not detect the error in the role rate filter constant software file.

Control Algorithm Flaws: QA verified only that the signoff report containing the load tape constants had all the proper signatures, an obviously inadequate process. This accident is indicative of the problems with QA as generally practiced and why it is often ineffective. The LMA Quality Assurance Plan used was a top-level document that focused on verification of process completion, not on how the processes were executed or implemented. It was based on the original General Dynamics Quality Assurance Plan with recent updates to ensure compliance with ISO 9001. According to this plan, the LMA Software Quality Assurance staff was required only to verify that the signoff report containing the constants had all the proper signatures; they left the I1 constant generation and validation process to the flight software and control dynamics engineers. Software Quality Assurance involvement was limited to verification of software checksums and placing quality assurance stamps on the software products that were produced.

2.8 Developer Testing Process

Once the error was introduced into the load tape, it could potentially have been detected during verification and validation. Why did the very comprehensive and thorough developer and independent verification and validation process miss this error?

Safety Constraint Violated: Testing must be performed on the as-flown software (including load tape constants).

Flaws in the Testing Process: The INU (FCS and IMS) was never tested using the actual constants on the load tape:

- Honeywell wrote and tested the IMS software, but they did not have the actual load tape.
- The LMA Flight Analogous Simulation Test (FAST) lab was responsible for system test, i.e., they tested the compatibility and functionality of the flight control software and the Honeywell IMS. But the FAST lab testing used a 300 Hertz filter simulation data file for IMS filters and not the flight tape values. The simulation data file was built from the original, correctly specified values of the designed constants (specified by the LMA CS engineer), not those entered by the software personnel in the generation of the flight load tape. Thus the mix of actual flight software and simulated filters used in the FAST testing did not contain the I1(25) error, and the error could not be detected by the internal LMA testing.

Process Model Mismatch: The testing capability that the current personnel thought the lab had did not match the real capability. The LMA FAST facility was used predominantly to test flight control software developed by LMA. The lab had been originally constructed with the capability to exercise the actual flight values for the I1 roll rate filter constants, but that capability was not widely known by the current FAST software engineers until after this accident;

knowledge of the capability had been lost in the corporate consolidation/evolution process so the current software engineers used a set of default roll rate filter constants. Later it was determined that had they used the actual flight values in their simulations prior to launch, they would have caught the error.

2.9 Independent Verification and Validation (IV&V)

Safety Constraint Violated: IV&V must be performed on the as-flown software and constants. All safety-critical data and software must be included in the IV&V process.

Dysfunctional Interactions: Each component of the IV&V process performed its function correctly, but the overall design of the process was flawed. In fact, it was designed in such a way that it was not capable of detecting the error in the role rate filter constant.

Analex was responsible for the overall IV&V effort of the flight software. In addition to designing the IV&V process, Analex-Denver performed the IV&V of the flight software to ensure the autopilot design was properly implemented in the software while Analex-Cleveland verified the design of the autopilot but not its implementation. The "truth baseline" provided by LMA, per agreement between LMA and Analex, was generated from the constants verified in the Signoff Report.

In testing the flight software implementation, Analex-Denver used IMS default values instead of the actual I1 constants contained on the flight tape. Generic or default I1 constants were used because they believed the actual I1 constants could not be adequately validated in their rigid body simulations, i.e., the rigid body simulation of the vehicle would not exercise the filters sufficiently[5]. They found out after the mission failure that had they used the actual I1 constants in their simulation, they would have found the order of magnitude error.

Analex-Denver also performed a range check of the program constants and the Class I flight constants and verified that format conversions were done correctly. However the process did not require Analex-Denver to check the accuracy of the numbers in the truth baseline, only to do a range check and a bit-to-bit comparison against the firing tables, which contained the wrong constant. Thus the format conversions they performed simply compared the incorrect I1(25) value in the firing tables to the incorrect I1(25) value after the conversion, and they matched. They did not verify that the designed I1 filter constants were the ones actually used on the flight tape.

Analex-Cleveland had responsibility for verifying the functionality of the design constant but not the actual constant loaded into the Centaur for flight. That is, they were validating the design only and not the "implementation" of the design. Analex-Cleveland received the Flight Dynamics and Control Analysis Report (FDACAR) containing the correct value for the roll filter constant. Their function was to validate the autopilot design values provided in the FDACAR. That does not include IV&V of the I1 constants in the flight format. The original design work was correctly represented by the constants in the FDACAR. In other words, the filter constant in question was listed in the FDACAR with its correct value of -1.992476, and not the value on the flight tape (-0.1992476).

Control Algorithm Flaws: Analex developed (with LMA and government approval) an IV&V program that did not verify or validate the I1 filter rate constants actually used in flight.

[5]Note that almost identical words were used in the Ariane 501 accident report [15].

The I1 constants file was not sent to Analex-Cleveland for autopilot validation because Analex-Cleveland only performed design validation. Analex-Denver used default values for testing and never validated the actual I1 constants used in flight.

Process Model Mismatches: The decision to use default values for testing (both by LMA FAST lab and by Analex-Denver) was based on a misunderstanding about the development and test environment and what was capable of being tested. Both the LMA FAST lab and Analex-Denver could have used the real load tape values, but did not think they could.

In addition, Analex-Denver, in designing the IV&V process, did not understand the generation or internal verification process for all the constants in the "truth baseline" provided to them by LMA. The Analex-Denver engineers were not aware that the I1 filter rate values provided originated from a manual input and might not be the same as those subjected to independent V&V by Analex-Cleveland.

None of the participants was aware that nobody was testing the software with the actual load tape values nor that the default values they used did not match the real values.

Coordination: This was a classic case of coordination problems. Responsibility was diffused among the various partners, without complete coverage. In the end, nobody tested the load tape—everyone thought someone else was doing it.

2.10 Systems Engineering

System engineering at LMA was responsible for the identification and allocation of the functionality to be included in the system. In fact, the software filter involved in the loss was not needed and should have been left out instead of being retained, yet another example of asynchronous evolution. Why was that decision made? The filter was designed to prevent the Centaur from responding to the effects of Milstar fuel sloshing and inducing roll rate errors at 4 radians/second. Early in the design phase of the first Milstar satellite, the manufacturer asked to filter that frequency. The satellite manufacturer subsequently determined filtering was not required at that frequency and informed LMA. However, LMA decided to leave the filter in place for the first and subsequent Milstar flights for consistency.[6] No further explanation is included in the report.

2.11 LMA Project Management (as Prime Contractor)

Safety Constraint: Effective software development processes must be established and monitored. System safety processes must be created to identify and manage system hazards.

Context: The Centaur software process was developed early in the Titan/Centaur program: Many of the individuals who designed the original process were no longer involved in it due to corporate mergers and restructuring (e.g., Lockheed, Martin Marietta, General Dynamics) and the maturation and completion of the Titan IV design and development. Much of the system and process history and design rationale was lost with their departure.

[6] This factor is similar to the Ariane 501 loss, where unnecessary software code was left in for "consistency" [15].

Control Algorithm Flaws:

- A flawed software development process was designed. For example, no process was provided for creating and validating the flight constants.
- LMA, as prime contractor, did not exert adequate control over the development process. The Accident Investigation Board could not identify a single process owner responsible for understanding, designing, documenting, or controlling configuration and ensuring proper execution of the process.
- An effective system safety program was not created.
- An inadequate IV&V program (designed by Analex-Denver) was approved and instituted that did not verify or validate the I1 filter rate constants used in flight.

Mental Model Flaws: Nobody seemed to understand the overall software development process and apparently all had a misunderstanding about the coverage of the testing process.

2.12 Defense Contract Management Command (DCMC)

Control Inadequacies: The report is vague about the role of DCMC, saying only that it was responsible for contract administration, software surveillance, and overseeing the development process. It does say that DCMC approved an IV&V process with incomplete coverage and that there was a software quality assurance function operating at DCMC, but it operated without a detailed understanding of the overall process or program and therefore was ineffective.

Coordination: No information was provided in the accident report although coordination problems between SMC and DCMA may have been involved. Were each assuming the other was monitoring the overall process? What role did Aerospace Corporation play? Were there gaps in the responsibilities assigned to each of the many groups providing oversight here? How did the overlapping responsibilities fit together? What kind of feedback did DCMC use to perform their process monitoring?

2.13 Air Force (Program Office): Space and Missile Systems Center Launch Directorate (SMC)

Safety Constraint: SMC must ensure that the prime contractors creates an effective development and safety assurance program.

Context: Like 3SLS, the Air Force Space and Missile System Center Launch Directorate was transitioning from a task oversight to a process insight role and had, at the same time, undergone personnel reductions.

Control Algorithm Flaws:

- The SMC Launch Programs Directorate essentially had no personnel assigned to monitor or provide insight into the generation and verification of the software development process. The Program Office did have support from Aerospace to monitor the software development and test process, but that support had been cut by over 50 percent since 1994. The Titan Program Office had no permanently assigned civil service or military personnel nor

full-time support to work the Titan/Centaur software. They decided that because the Titan/Centaur software was "mature, stable, and had not experienced problems in the past" they could best use their resources to address hardware issues.

- The transition from oversight to insight was not managed by a detailed plan. AF responsibilities under the insight concept had not been well defined, and requirements to perform those responsibilities had not been communicated to the workforce. In addition, implementation of the transition from an oversight role to an insight role was negatively affected by the lack of documentation and understanding of the software development and testing process. Similar flawed transitions to an "insight" role are a common factor in many recent aerospace accidents.

- The Titan Program Office did not impose any standards (e.g., Mil-Std-882) or process for safety. While one could argue about what particular safety standards and program could or should be imposed, it is clear from the complete lack of such a program that no guidance was provided. Effective control of safety requires that responsibility for safety be assigned at each level of the control structure. Eliminating this control leads to accidents. The report does not say whether responsibility for controlling safety was retained at the program office or whether it had been delegated to the prime contractor. But even if it had been delegated to LMA, the program office must provide overall leadership and monitoring of the effectiveness of the efforts. Clearly there was an inadequate safety program in this development and deployment project. Responsibility for detecting this omission lies with the program office.

2.14 Summary

Understanding why this accident occurred and making the changes necessary to prevent future accidents requires more than simply identifying the proximate cause—a human error in transcribing long strings of digits. This type of error is well known and there should have been controls established throughout the process to detect and fix it, including identifying the roll rate filter data as critical through a hazard analysis process. Either these controls were missing in the development and operations processes or they were inadequately designed and executed.

Even though this accident report was unusual in its depth of analysis of the causal factors, the STAMP analysis identified several additional questions that might have been asked to provide additional insight into how such accidents might be avoided in the future, particularly at the higher levels of the development and operations control structures. Identification of coordination problems was a common and important omission; for example, were the various process auditors (DCMC, Aerospace Corporation, QA, etc.) all assuming that someone else was monitoring the process?

The result of a STAMP accident analysis is an understanding of the role each component played in the accident process. This information (summarized in Figures 9 and 10) provides guidance in creating recommendations for preventing future accidents. In the case of this accident, better development and verification, system safety engineering, management, and oversight processes are needed. For example, looking at the Prime Contractor, recommendations for redesign of the IV&V process, better specified processes for load tape creation, design of a comprehensive system safety program that identifies hazards and critical data and then institutes special controls for them, and more oversight and monitoring of the software development process seem appropriate. Operations management needs to clarify launch personnel roles, to

provide tools and information for monitoring software behavior at the launch site, create better problem reporting and resolution processes, augment personnel training programs, etc.

In addition to investigating and analyzing accidents, a systems-theoretic accident model can be used to prevent accidents and to accumulate the information necessary to design for safety during system design and development. Hazard analysis is essentially the investigation of an accident before it occurs. A proactive accident investigation, i.e., hazard analysis, using STAMP rather than the traditional analysis techniques based on event-chain models (e.g., fault tree analysis, event tree analysis, and failure modes and effects criticality analysis) can provide the information necessary to design an integrated socio-technical system, including development and operations, to prevent accidents in software-intensive systems [6].

3 Conclusions

Accident models provide the basis for safety engineering—both in the analysis of accidents that have occurred and in the development of techniques to prevent accidents. This paper has suggested that dealing with safety in software-intensive systems will require more sophisticated accident models than those currently in use, which were developed for electromechanical systems. A proposal was made that models based on systems theory would be appropriate. One such model was described and illustrated using a software-related accident. Other types of models are possible.

A major difference between a systems-theoretic accident model and a chain-of-events model is that the former does not identify a root cause of an accident. Instead, the entire safety control structure is examined and the accident process to determine what role each part of the process played in the loss. While perhaps less satisfying in terms of assigning blame, the systems-theoretic analysis provides more information in terms of how to prevent future accidents.

References

[1] Russell L. Ackoff. Towards a system of systems concepts. *Management Science*, 17(11):661–671, July 1971.

[2] Ashby, W.R., 1956. *An Introduction to Cybernetics*. Chapman and Hall, London.

[3] Bertalanffy, L. *General Systems Theory: Foundations, Development, and Applications*. G. Braziller, New York, 1969.

[4] Peter Checkland. *Systems Thinking, Systems Practice*. John Wiley & Sons, New York, 1981.

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

[6] Nicolas Dulac and Nancy Leveson. An approach to design for safety in complex systems. *Int. Symposium on Systems Engineering (INCOSE)*, Toulouse, France, June 2004,

[7] Harold Gehman (Chair). Columbia Accident Investigation Report. August 2003.

[8] Institute of Electrical and Electronics Engineers. *IEEE Standard Computer Dictionary*. New York, NY 1990.

[9] Jacques Leplat. Occupational accident research and systems approach. In Jens Rasmussen, Keith Duncan, and Jacques Leplat, editors, *New Technology and Human Error*, pages 181–191, John Wiley & Sons, New York, 1987.

[10] Nancy G. Leveson. *Safeware: System Safety and Computers.* Addison-Wesley, 1995.

[11] Nancy G. Leveson. A New Accident Model for Engineering Safer Systems. *Safety Science*, Vol. 42, No. 4, Elsevier, April 2004.

[12] Nancy G. Leveson. The role of software in spacecraft accidents. AIAA Journal of Spacecraft and Rockets, July 2004.

[13] Nancy G. Leveson. A New Approach to System Safety Engineering. Manuscript in preparation, draft can be viewed at http://sunnyday.mit.edu/book2.pdf.

[14] Nancy Leveson, Mirna Daouk, Nicolas Dulac, and Karen Marais. Applying STAMP in Accident Analysis. *2nd Workshop on the Investigation and Reporting of Accidents*, Williamsburg, Virginia, September, 2003.

[15] J.L. Lions. Ariane 501 Failure: Report by the Inquiry Board. European Space Agency, 19 July 1996.

[16] Ralph F. Miles Jr. Introduction. In Ralph F. Miles Jr., editor, *Systems Concepts: Lectures on Contemporary Approaches to Systems*, pages 1–12, John F. Wiley & Sons, New York, 1973.

[17] NASA/ESA Investigation Board. SOHO Mission Interruption. NASA, 31 August 1998.

[18] Pavlovich, J.G. Formal Report of Investigation of the 30 April 1999 Titan IV B/Centaur TC-14/Milstar-3 (B-32) Space Launch Mishap. U.S. Air Force, 1999.

[19] Charles Perrow. *Normal Accidents.* Basic Books, 1984 (republished by Princeton University Press, 1999).

[20] Joan L. Piper. *Chain of Events: The Government Cover-Up of the Black Hawk Incident and the Friendly Fire Death of Lt. Laura Piper.* Brasseys Inc., 2001.

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

[22] Jens Rasmussen. Risk Management in a Dynamic Society: A Modelling Problem. *Safety Science*, vol. 27, No. 2/3, Elsevier Science Ltd., 1997, pages 183-213.

[23] Jens Rasmussen and Inge Svedung. *Proactive Risk Management in a Dynamic Society*, Swedish Rescue Services Agency, 2000.

[24] Jens Rasmussen. Human error and the problem of causality in analysis of accidents. In D.E. Broadbent, J. Reason, and A. Baddeley, editors, *Human Factors in Hazardous Situations*, pages 1–12, Clarendon Press, Oxford, 1990.

[25] Jens Rasmussen, Annelise Mark Pejtersen, and L.P. Goodstein. *Cognitive System Engineering.* John Wiley & Sons, 1994.

[26] William P. Rogers. *Report of the Presidential Commission on the Space Shuttle Challenger Accident*. U.S. Government Accounting Office, Washington, D.C., 1986.

[27] U.S. Government Accounting Office, Office of Special Investigations. Operation Provide Comfort: Review of Air Force Investigation of Black Hawk Fratricide Incident (GAO/T-OSI-98-13). U.S. Government Printing Office, Washington, D.C. 1997.

[28] Gerald Weinberg. *An Introduction to General Systems Thinking*. John Wiley & Sons, New York, 1975.

[29] Norbert Weiner. Cybernetics: or the Control and Communication in the Animal and the Machine. The MIT Press, 2nd Edition, 1965.

[30] David D. Woods. Lessons from beyond human error: Designing for resilience in the face of change and surprise. Design for Safety Workshop, NASA Ames Research Center, October 8-10, 2000.