JOURNAL OF AEROSPACE INFORMATION SYSTEMS
Vol. 11, No. 6, June 2014

# Improving Hazard Analysis and Certification of Integrated Modular Avionics

Cody Harrison Fleming* and Nancy G. Leveson†
*Massachusetts Institute of Technology, Cambridge, Massachusetts 02139*

DOI: 10.2514/1.I010164

**Integrated modular avionics systems present new opportunities and benefits for developing advanced aircraft avionics, as well as a series of challenges related to hazard analysis and certification. This paper addresses some of those challenges and proposes a new procedure for improving hazard analysis of integrated modular avionics systems. A significant objective of integrated modular avionics architectures is the ability to develop individual software applications independently and then integrate those applications onto one platform. It has been very difficult for both designers and certifiers to understand and predict how the system will behave when the applications are integrated into one system. Traditional fault-based hazard analysis techniques are limited with respect to this problem. Therefore, this paper uses a different technique, called Systems-theoretic Process Analysis, to identify hazardous behavior that emerges when individual applications are integrated. Systems-theoretic process analysis is a systems-theoretic hazard analysis technique that accounts for hazardous behavior due to component interaction, including cases when the components have not failed or faulted. Systems-theoretic process analysis is extended in this paper to account for behavior that emerges when software applications share data, which is a requirement in aircraft systems. The paper illustrates the new approach with an example that includes real-world avionics functions.**



## I. Introduction

INTEGRATED modular avionics, or IMA, "is a shared set of flexible, reusable, and interoperable hardware and software resources that, when integrated, form a platform that provides services, designed and verified to a defined set of safety and performance requirements, to host applications performing aircraft functions" [1]. By putting multiple applications onto one platform, IMA architectures yield savings in space, weight, power, and maintenance infrastructure.

However, due to the lack of physical infrastructure and the design constraints associated with architectures where individual avionics functions must be physically connected, there is a concern that the use of IMA will result in a rapid increase in system complexity, unintended interaction amongst components, and integration responsibilities for the IMA developer. The existing regulatory regime, which was developed for traditional federated architectures, is considered to be inadequate for the predicted future explosion in complexity [2,3]. A vital component of the regulatory approach is the safety assessment, and this too is inadequate [4]:

> Traditional safety assessment guidance such as Society of Automotive Engineers (SAE) Aerospace Recommended Practice (ARP) 4754 [5] and SAE ARP 4761 [6] are being applied to IMA.
> The existing ARP guidance is well suited for an individual system tied directly to an aircraft function. However, the existing safety assessment guidance lacks focus on the integration aspects of an IMA system. As opposed to traditional avionics, IMA's integration of multiple functions with shared resources requires the system safety assessment (SSA) process to focus more on common mode, common cause, and cascading effects in addressing the integration of functions and the multiple loss [sic] of functions. A different approach is needed.

In addition, because independent development of applications is an attractive potential benefit of IMA, developers would like to modify or upgrade applications and airframe manufacturers may want to insert entirely new applications [7]. Therefore, a change impact analysis, particularly with respect to hazardous behavior, is also needed from a regulatory perspective [3].

This paper outlines a new approach for analyzing hazardous behavior due to interactions among functions in a complex avionics suite such as IMA. In addition to a hazard analysis technique for IMA, this paper also introduces a change impact hazard analysis that captures hazardous behavior due to changes in application behavior, data sharing between applications, and other changes such as mechanical or hardware modifications. The method potentially reduces the amount of rework required for certification of a modified or inserted application. This paper first introduces the concepts and underlying assumptions of IMA, introduces a new approach for safety analysis that could aid in the certification of IMA systems, and then describes the approach using an example IMA implementation.

## II. Background

To understand the new analysis approach being suggested, some basic information about integrated modular avionics, the current regulatory regime, and the limitations of the current techniques is helpful.

Received 14 August 2013; revision received 20 December 2013; accepted for publication 23 January 2014; published online 6 June 2014. 

*Ph.D. Candidate, Department of Aeronautics and Astronautics; chf44@mit.edu.
†Professor, Department of Aeronautics and Astronautics and Engineering Systems Division.

### A. IMA Architectures

Traditional, federated avionics architectures have distributed, self-contained, and dedicated computing, communication, and input/output services. In contrast, IMA involves the execution of multiple applications on a common hardware resource. Figure 1 compares the two architectural concepts [8].

By replacing numerous separate processors and line replaceable units with fewer, more centralized processing units, IMA architectures promise significant weight reduction, reduced power consumption, and maintenance savings for modern aircraft systems [9]. In addition to weight savings and power reduction, IMA considerably reduces the amount of required wiring and the attendant burden put on airframe designers and maintenance staff.

Another attractive benefit of the modular approach is the potential for independent development of software applications by distinct groups across organizational boundaries within a manufacturer or groups from entirely different companies or organizations.

### B. Regulatory Approach

The current regulatory approach for IMA systems is fundamentally similar to that used for traditional federated systems. For example, failure modes and effects analysis (FMEA) is typically used as a recommended practice for developing a system safety assessment [2]. FMEA focuses on individual component failures, which are the "inability of a component to perform its designed function within specified limits" [10].

The system safety assessment from the FMEA is used to guide application software developers in certain life-cycle processes. These processes include design reviews, coding practices, verification and testing, and integration, all of which are commensurate with the required level of safety derived from the FMEA [11].

A key requirement for designing IMA systems is the use of partitioning. Partitioning allocates time and memory to an application, allowing the IMA to host multiple applications with different responsibilities and to operate on shared resources. The shared resources include the real-time operating systems (RTOSs), central processing unit(s), memory management unit(s), and input/output handlers [12]. Partitioning, which is part of the development paradigm used in traditional federated architectures, necessarily becomes more important in an IMA system. Due to the emphasis on partitioning, an additional set of recommendations is put forth as part of IMA development, including considerations for the development of a "robust partition" (which is called robust to emphasize its importance in IMA, as compared to standard partitioning), as well as a common application interface for separate software developers [13].

In theory, robust partitioning facilitates the isolation of individual applications. This isolation then allows traditional hazard analysis techniques, such as FMEA, to focus on identifying potential faulted modes of these individual components and their effects on system performance.

### C. Integration and Interface Control

There are two key concepts necessary to ensure that application software can be developed independently: integration and interface control.

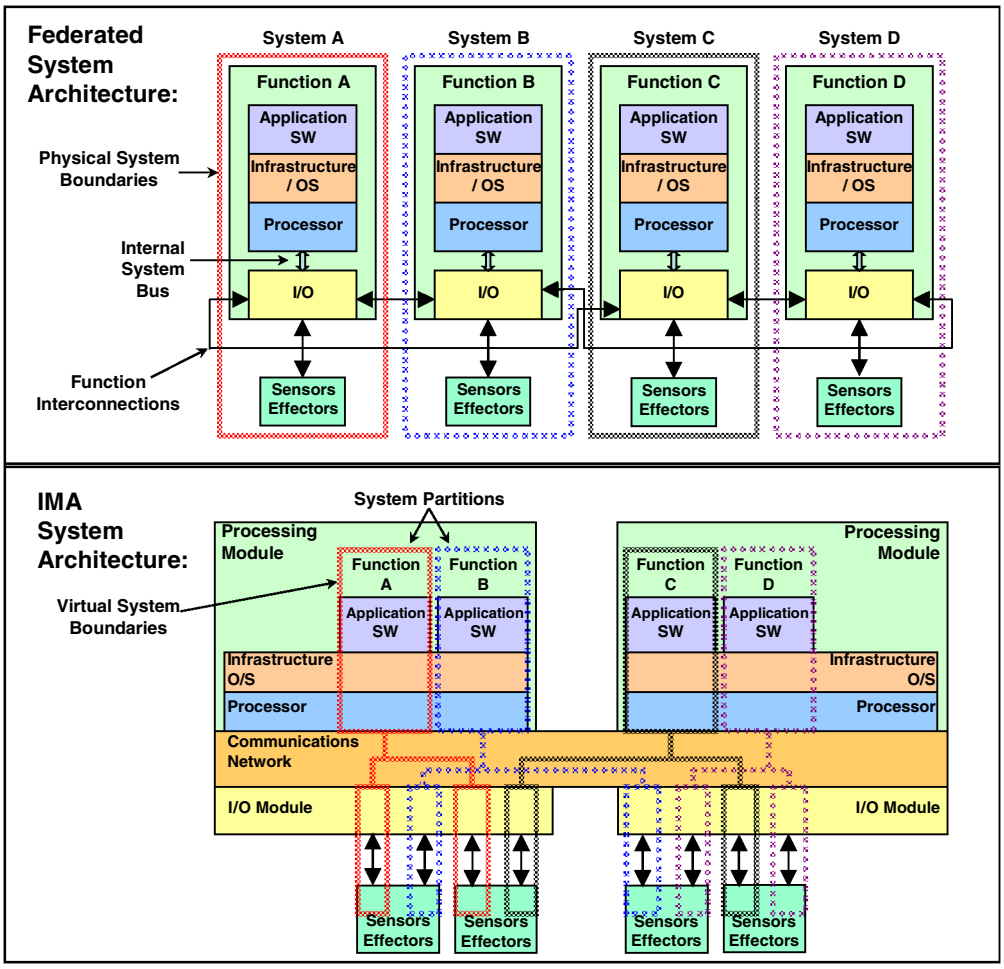



**Fig. 1 Comparison of federated and IMA architectures [8] (SW, software; OS, operating system).**

Individual developers must ensure that the software design adheres to an application programming interface (API), and the IMA system developer must ensure that all applications are integrated properly on the operating system [14]. Partitioning then limits the amount of interaction between applications. The number and type of applications that can run within a partition are limited by the constraints designed into the operating system, according to the partition management hardware and software typically developed by the IMA system integrator.

However, applications can and must share information (and, indeed, applications may share control commands in some instances), leading to the second requirement, which is interface control. Interfaces are typically specified, in an interface control document (ICD), for electrical interactions, mechanical or physical interactions, and software modules. A typical software interface specification would include the following items, according to [15]: 1) digital signal characteristics; 2) data transmission format, coding, timing, and updating requirements; 3) Data and data element definition; 4) message structure and flow; 5) operational sequence of events; and 6) error detection and recovery procedures.

In particular, a software interface should "define the actions taken to process the data and return the results of the process" [15]. The ICD approach emphasizes the instructions necessary to input or output data but does not explicitly prescribe the underlying logic or behavior that actually generates the variables that should go back onto the interface. (In fact, if the ICD did specify the underlying behavior of a software module or application, it should technically not be considered an ICD.)



### D. Limitations

There are significant assumptions embedded in relying on partitioning and interface control documents to assure safety in IMA. The first assumption is that robust partitioning, combined with some form of interface control, will result in the isolation of avionics functions. The second assumption is that interface control will lead to a thorough understanding of those interactions between functions that are not "eliminated" by partitioning. These assumptions do not hold in all cases, however, particularly when there is a high degree of data dependence and control dependence between avionics applications. The current approach is particularly limited with respect to understanding hazardous interactions among applications, designing the system and its applications to reflect this understanding, and certifying the resulting system [2–4,12,16].

To a certain degree, robust partitioning does help to achieve these aims, but data dependency is still a necessary and desirable property of avionics functions. There are two types of dependency involved here. "Data coupling" is the sharing of information both within and outside of a partition. A second type of coupling, "control coupling," exists when one software component influences the execution of another software component [17]. Despite the implementation of robust partitioning, coupling can still exist because aircraft avionics necessarily require information from other functions. In fact, due to the virtualization of interfaces between functions, future avionics systems may have even "more" coupling than the physical, federated systems of the past. The wiring and harness used to route data in traditional federated systems is certainly cumbersome and difficult to manage, but it also provides an intrinsic constraint on the degree of coupling between avionics. Virtualization of these physical interfaces eliminates this constraint.

In a survey outlining the gaps in Federal Aviation Administration (FAA) systems integration and safety assurance [2], uses the following incident. The incident provides an example of how important it is to thoroughly evaluate component coupling and cascading effects in order to properly assure safety, and how difficult this evaluation can become. The information comes from the serious incident investigation report of a B747 incident, issued by the South African Civil Aviation Authority [18]. The report offered the following executive summary of the incident:

> The serious incident involved the uncommanded retraction of the automatic Group 'A' leading edge flaps on rotation for a period of about 23 seconds. Subsequent to the initiation of the retraction of the Group 'A' leading edge flaps, the aircrew was faced with unexpected stall warnings. At no time was the aircrew aware that the Group 'A' leading edge flaps had retracted or as to the circumstances leading to the stall warnings. They were however aware that the thrust reverser in-transit EICAS amber message on the P2-Pilots Center Instruments Panel did display during takeoff roll prior to rotation . . . No evidence was found that the thrust reversers had in fact deployed.
>
> The automatic retraction of the Group 'A' leading edge flaps based on dual thrust reverser in-transit signals from either both inboard or both outboard engines was part of the original 747-400 type design. All model 747 airplanes will automatically retract the Group 'A' LE flaps upon movement of the reverse thrust handle. This is to prevent thrust reverser efflux air from impinging directly onto the flap panel surfaces in order to improve the fatigue life of the panels and their attachments. The original type design added the auto-retract feature based upon receipt of a thrust reverser unlock signal . . .
>
> [The occurrence was] caused by the automatic LE flap retraction logic retracting the Group 'A' LE flaps on receipt of spurious thrust reverser unlock signals from the no. 2 and no. 3 engines. The possibility of such an occurrence had not been identified during amendment of the retraction logic.

Note that the flight crew was able to keep the aircraft flying until the leading-edge (LE) flaps reextended. Figure 2 shows the sequence of events in simplified, graphical form. Bartley and Lingberg [3] observed that the secondary effects caused by failure of shared or coupled resources were not immediately obvious, and the preceding example of the B747 certainly corroborates that conclusion. These kinds of "unintended interactions" are not unique to IMA systems and happened on decades-old systems. However, with the virtualization of interfaces inherent in IMA architecture, systems with IMA will become much more complex, and the potential interaction between IMA functions and their virtual interfaces may increase by orders of magnitude compared to simpler, federated systems. The result of this increase in complexity is that many "secondary" effects may become even more obscured than what was already the case in the past. The methods currently being used to analyze these systems are not enough to understand and manage this complexity, and thus properly design and develop them.

One suggested approach to assess whether a change impact analysis is necessary is by determining whether an interface control document revision is necessary [3,15]. The assumption underlying the use of the ICD is that the separate functions are isolated from each other by robust partitioning and can only communicate via a very defined and specific mechanism that minimizes data and control coupling between partitions. Therefore, if a change to a certain function does not require a change to the ICD, which defines the output parameters from that function, then the functions using those parameters cannot be impacted by the change.

This assumption is false. Bartley and Lingberg [3] provided an example involving flaps control. Figure 3 shows the interface between the flaps system controller where a discrete variable of "flaps extended" is passed between supposedly independent avionics functions. Without further definition of the exact technical meaning of flaps extended, a true state of the flaps-extended discrete variable could possibly mean any of the following:

- Both left and right trailing-edge (TE) flap surfaces are detected in the 1 or greater flap detent.[‡]
- Both left and right trailing-edge flap surfaces are not detected in the "up" flap detent.

[‡]A detent is a device used to mechanically resist or arrest the rotation of a wheel, axle, or spindle.

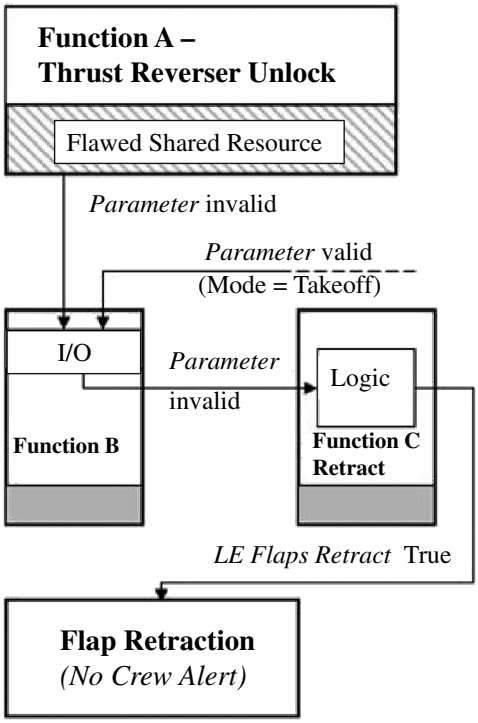



**Fig. 2 Simplified example of secondary effects of shared resource, adapted from [18].**

- The flap lever handle is detected in the 1 or greater flap handle detent.
- The flap lever handle is not detected in the up flap handle detent.

All four of these possibilities have differing characteristics from each other, and different receiving functions might use the flaps-extended discrete in a different way. For example, once the trailing-edge flaps begin to move, the logic that determines "flaps not in the up position" will be satisfied almost immediately. In contrast, the flaps may take 5 to 10 s to fully reach the "flaps detected in the '1' detent" position. Furthermore, if the flap surfaces will not respond to a valid command due to a hydraulic system failure, the flap lever position will no longer reflect the true position of the flap surfaces once the lever is moved out of the up detent. Obviously, how a signal is computed needs to be understood by the designers of any function that uses that particular signal. Differences such as those described may be of critical importance to using the system.

If only the information in Fig. 3 is communicated via the data bus, then a change in the logic of the flap system controller may negatively impact the behavior of the function using the flaps-extended variable. The receiving system(s) developers will thus not be privy to the change, and the system-level impact will not be understood. For example, a problem discovered later in the development cycle, after the design of the individual applications have been "finalized," may lead to a change in the logic of the flap system controller. One potential method of addressing this problem with the flap alert logic is to compute the flaps-extended variable using the flap lever position instead of the actual flap surface position. The physical meaning of the flaps-extended variable has changed.

The following is according to [3]:

> The point illustrated is that it cannot not [sic] be left to the designers and system experts of the function being updated (in this example, the flap system) to determine the effect of that change on downstream users. The experts that are required to assess the impact of the change are the designers and analysts of the using system, not the source system. A Change Impact Analysis that crosses functional boundaries is

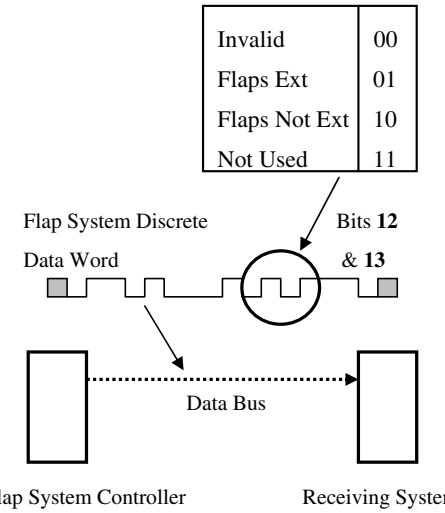



**Fig. 3 Flaps-extended (Ext) discrete data transmission [3].**

clearly required for this example, even though the ICD for the source function did not change. Additionally, this example illustrates why robust partitioning does not totally insulate functions residing in one partition from changes to a function in a different partition.

Some organizations suggest the use of an interface definition document, which is a unilateral document controlled by the end-item provider, where the user has no input into what should (or should not) be specified on the interface [19]. This approach merely exacerbates some of the issues outlined in the previous flaps-extended example.

Robust partitioning and interface control documents are clearly valuable and necessary aspects of IMA system development. However, as the aforementioned examples illustrate, there must be additional tools to aide in the design, integration, and certification of these systems.

Traditional hazard analysis techniques such as FMEA and fault tree analysis focus on individual component failure or faulted modes. This focus on component reliability assumes that robust partitioning and interface control are valid and that components do not interact either directly or indirectly. However, as the previous examples illustrate, these assumptions usually do not hold for complex systems.

### E. Related Research

While the regulatory, industry, and research communities recognize the differences between traditional avionics architectures and IMA, there has been relatively little research directly focused on hazard analysis and certification of IMA systems. Conmy and McDermid proposed a failure analysis technique driven by guidewords related to generic IMA functions [20]. Guidewords such as omission, commission, early, late, or value are used to generate deviations from expected behavior for each IMA function. For each deviation, a set of causes is then generated and associated mitigation strategies are generated. Furthermore, Conmy and McDermid developed a contracts-based approach where interdependent functions supply behavioral "contracts" to user functions. [21]. The Conmy and McDermid approach focuses on failure or faulted modes of operation only. They do not handle hazardous behavior resulting from interactions of functions that are behaving nominally, such as in the flaps controller example.

Rushby suggested a composition framework for certification of modular architectures [12]. Such a framework should have three properties, even in the presence of faults: composability, where properties of components or prior compositions are preserved when new components are added; compositionality, where system properties are derived solely from component properties; and monotonicity, system properties are not reduced when a component is replaced by a superior one having more properties. Rushby suggested that formal methods based on infinite bounded model checking may provide the necessary capability for analyzing such properties. The three properties of Rushby's composition framework may very well contribute to an improvement in the modular software development paradigm. However, the composition framework may violate a fundamental principle of systems theory, which is "emergence." Emergence is the principle that whole entities exhibit properties that are meaningful only when attributed to the whole, and not to its parts [22]. Leveson, and others, has argued that system safety is an emergent property rather than a property of individual component behavior [23].

Yong et al. proposed an incremental certification methodology where different modules can be added to the certification artifacts as the design matures [24], and Ruiz et al. proposed a case-based reasoning approach for software reuse [25]. Ruiz et al., in particular, presents a rigorous and concise method for documenting the introduction of new data and for retrieving existing data. Neither Yong et al. [24] nor Ruiz et al. [25], however, suggested appropriate ways to perform the hazard analyses that lead to the artifacts in their methodologies.

The hazard and operability study (HAZOP), which is used in the process industries, is simply a different form of failure analysis, where failures are defined as deviations from nominal behavior [26]. Most of the attempts to apply HAZOP to software [27,28] are simply a form of FMEA and have not been successful [29].

While the hazard analysis techniques proposed in Sec. III have been used to analyze very large, complex systems both in aerospace and other domains [30–34], they have limited application to IMA systems. Suo et al. has developed an intent specification approach that uses the same hazard analysis techniques proposed in this paper [35]. Suo et al. generates a set of unsafe control actions for aspects of the IMA operating system and then develops a set of hierarchical safety requirements based on those unsafe control actions. Suo et al.'s work does not, however, focus on analyzing aircraft functions that operate on the IMA platform.

All of these approaches either focus on failures or faults of IMA components, do not pertain to hazard analysis of complex avionics systems, or do not deal with unforeseen behavior due to coupling of nominal component behavior. Our approach differs in that it looks beyond the underlying components of an IMA system, such as the operating system and health monitor. Of course, these are vital aspects of IMA, but we are interested in analyzing the applications that run on the IMA platform. Of particular importance is the ability to capture hazardous behavior due to interactions between IMA (or other aircraft) functions and the introduction of hazardous behavior due to changes in the behavior of those functions or the coupling between them.

## III. Proposed Approach

Our approach is based on a new accident causality model called the systems-theoretic accident model and processes (STAMP), which extends the types of accidents and causes that can be considered by including nonlinear, indirect, and feedback relationships among events [23]. STAMP extends the traditional chain-of-failure events causality model to include new types of accident causes arising from component interactions (rather than just component failures), cognitively complex human mistakes, software errors, requirements errors, etc. Accidents or unacceptable losses can result not only from system component failures but also from interactions among system components, both physical and social, that violate system safety constraints.

In systems theory, emergent properties (like safety) associated with a set of components are related to constraints upon the degrees of freedom of those components' behavior [22]. System safety, then, can be reformulated as a system control problem rather than a component reliability problem: accidents or losses occur when component failures, external disturbances, and/or dysfunctional interactions among system components are not handled adequately or controlled: where controls may be managerial, organizational, physical, operational, or manufacturing.

In a systems-theoretic view of safety, emergent behaviors 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 nonhazardous or safe system states: for example, the power must never be on when the access door to the high-power source is open; two aircraft must never violate minimum separation requirements; pilots in a combat zone must be able to identify targets as hostile or friendly; and the public health system must prevent the exposure of the public to contaminated water and food products. Accidents result from interactions among system components that violate these constraints: in other words, from a lack of appropriate constraints on component and system behavior.

### A. Systems-Theoretic Process Analysis

System-theoretic process analysis is a hazard analysis technique built on STAMP. It is used in this paper to identify the system constraints necessary to ensure safe development and operation of IMA systems. As described previously, accidents are viewed in STAMP as resulting from

inadequate enforcement of constraints on system behavior. STPA has three components: system-level analysis, identification of potentially unsafe control actions (called STPA step 1), and performing causal analysis on unsafe control actions from previous step (called STPA step 2). The rest of the section describes each step in greater detail.

STPA uses system hazards, safety constraints, and a functional control diagram. A hazard is a system state or set of conditions that, together with a particular set of worst-case environmental conditions, will lead to an accident or loss.

An example hazard for aviation is "two aircraft violate minimum separation." Related safety constraints for this hazard might include "air traffic controller must provide advisories that maintain safe separation between aircraft" or "warnings must be provided to flight crews when separation is violated." Both the hazards and safety constraints are refined and allocated to each component of the system.

Safety constraints are enforced through a hierarchical control structure. Figure 4 shows a generic functional control structure for a sociotechnical system. (The general control structure includes several levels of management and government oversight, but the analysis in the following section only considers the lower levels of operation.) Each hierarchical level of the control structure represents a control process and control loop with actions and feedback, and each level imposes constraints on the activity of the level beneath it. While classic component failures may lead to violation of the constraints, violations can also result from unsafe interactions among components operating as designed where software or human controllers issue unsafe control actions.

Identifying the potentially unsafe control actions for the specific system being considered is the first step in STPA. There are four types of hazardous control actions that need to be eliminated or controlled to prevent accidents:

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

The case must also be considered where a control action required for safety is provided but is not followed (executed).

These unsafe control actions are used to create safety requirements and constraints on the behavior of both the system and its components. Additional analysis can then be performed to identify the detailed scenarios leading to the violation of the safety constraints and used to generate more detailed safety requirements. As in any hazard analysis, these scenarios are the basis for designing controls and mitigation measures for the hazards. Any hazards that cannot be adequately controlled at the system level must be allocated in the form of behavioral requirements on the lower-level system components.

Human and automated controllers use a process model (usually called a mental model for humans) to determine what control actions are needed. The process model contains the controller's understanding of 1) the current state of the controlled process, 2) the desired state of the controlled process, and 3) the ways the process can change state. Software and human errors often result from incorrect process models; for example, the software thinks the spacecraft has landed and shuts off the descent engines. Accidents can therefore occur when an incorrect or

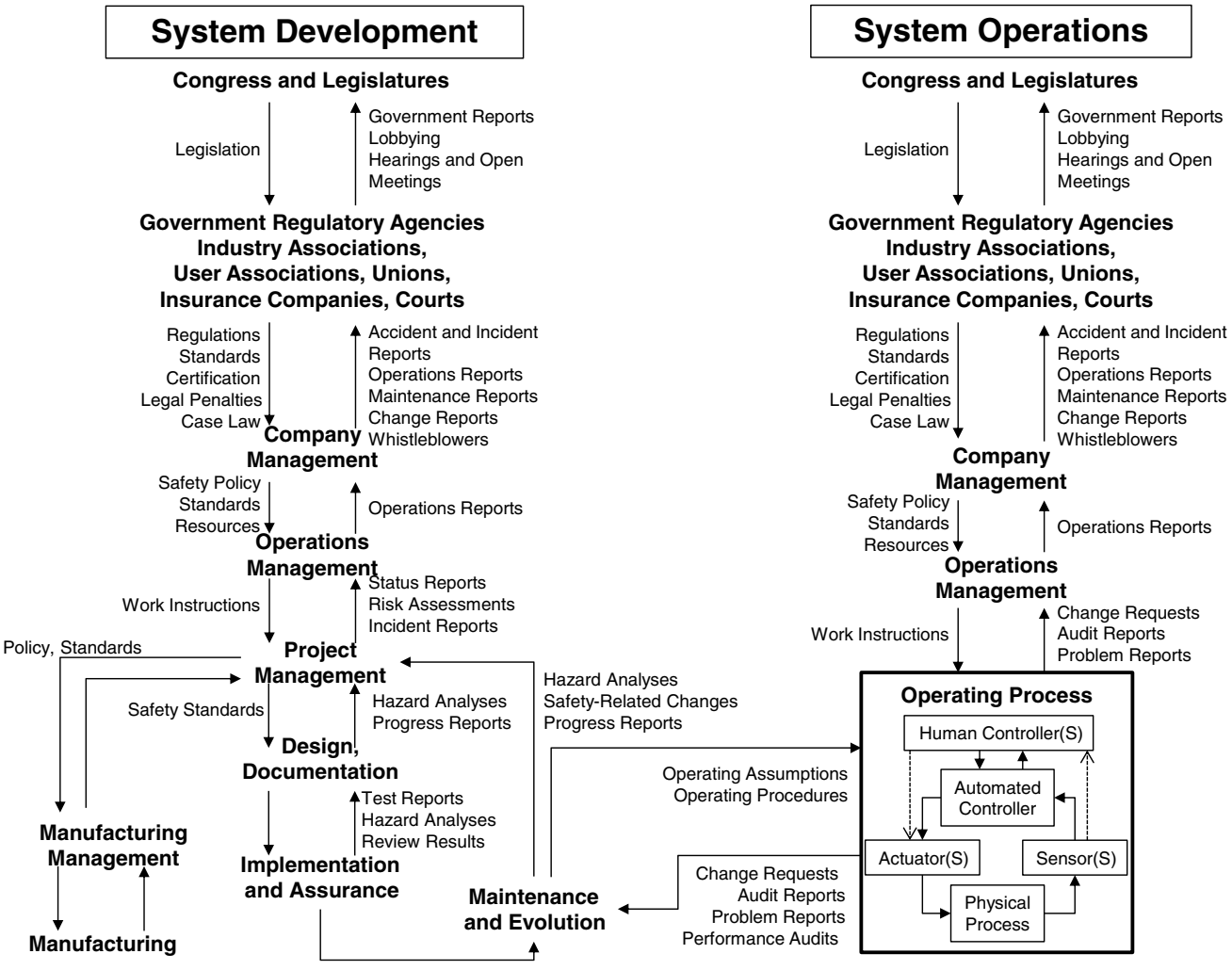



**Fig. 4 General sociotechnical control structure.**

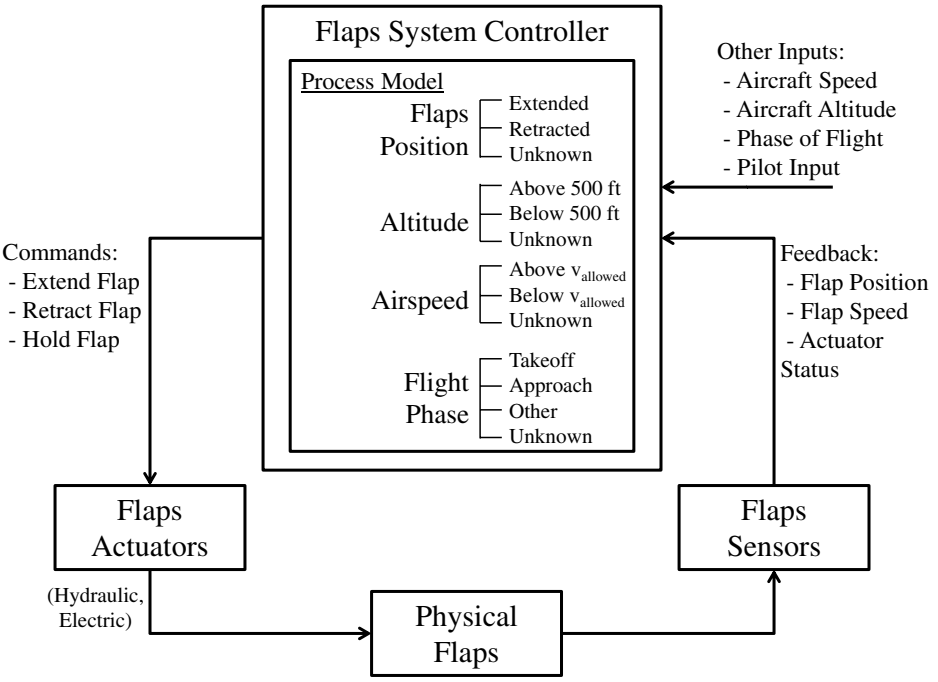



**Fig. 5 Example process model, flaps control.**

incomplete process model causes a controller to provide control actions that are hazardous. While process model flaws are not the only causes of accidents involving software and human errors, they are a major contributor to hazardous behavior when software or humans are involved.

Figure 5 shows a control loop for a simple model of the flaps controller. Observe the "Process Model" box within the controller. The process model contains the information necessary to make safe control actions, and if any of this information is inaccurate or if the algorithm for generating actions is flawed, the controller may generate unsafe commands. The process model contains several "process model variables" (for example, flap position and altitude), and these process model variables can take on different values listed in the figure.

The next step in STPA is to determine how each potentially hazardous control action identified in the previous step can occur. The generic control loop shown in Fig. 6 is used to generate these scenarios. The specific causal factors are identified using the general causal factors in the figure, examining each part of the loop for a specific controller and controlled process and using the relevant existing design details of that loop. By identifying the causes shown on the control loop, the analysis captures typical hardware failures but also helps identify flawed requirements in maintaining accurate and consistent process models.

The rest of this paper will use these important notions of process model, process model variables, system states, and the context in which control actions are generated.

### B. Global Process Model© Variable

The previous section described the importance of controllers' process models and process model variable states, and it outlined the fact that initializing and maintaining an accurate process model is vital to ensuring safe behavior. Section II described data coupling and the potential for greater coupling in IMA systems. Increasingly, a controller that performs a specific function will have a model of part of the system state that is also used by other controllers that perform different functions. For example, both the ground proximity warning system and flight management system need to have a model of the aircraft altitude, but they have very different responsibilities. In Fig. 7, there are $n$ processes, and each process has different safety-related constraints. Each controller has a model of the same component of the controlled system state: "State $i$.2."

Again, in the general STPA framework, an important contributor to hazardous behavior is an incorrect or inconsistent process model. This has typically meant that the controller's process model is inconsistent with its own controlled process. However, separate controllers requiring the same controlled system variables lead to a different class of process model consistencies. One controller's perception of a component of the controlled system state might be quite different than the definition of another controller. The flaps-extended state in Fig. 3 illustrates how different controllers may have very different conceptions of the same state.

Therefore, we introduce the concept of a Global Process Model variable (GPMV). By identifying data coupling at a higher level of abstraction than at an individual control loop, STPA will be able to identify the types of hazardous behavior outlined in Sec. II, such as component interaction due to cascading effects and change impact. These hazards are due to the fact that safety is an emergent system property, and the hazardous conditions arise when multiple software applications use and manipulate a shared set of variables.

### C. Procedure

The new procedure for analyzing hazardous behavior due to component interaction in an IMA system is outlined in Fig. 8. The procedure involves four types of analysees: 1) hazard analysis, 2) independence analysis, 3) coupling safety assessment, and 4) change impact analysis.

#### *1. Hazard Analysis Using STPA*

First, STPA is performed as described in Sec. III.A. The analysts develop a functional control structure and safety-related responsibilities based on the system architecture and high-level hazards. Then, control actions are identified for each controller and the analysis generates the potential unsafe control actions using the four general types as a guide (see Sec. III.A). Finally, each control loop is analyzed individually for causes of unsafe control.

Figure 9 shows the control structure of an IMA system that includes a flaps system controller along with a thrust reverse controller and flight deck display. The flaps system controller is responsible for controlling the position of the leading- and trailing-edge flaps, and the thrust reverse

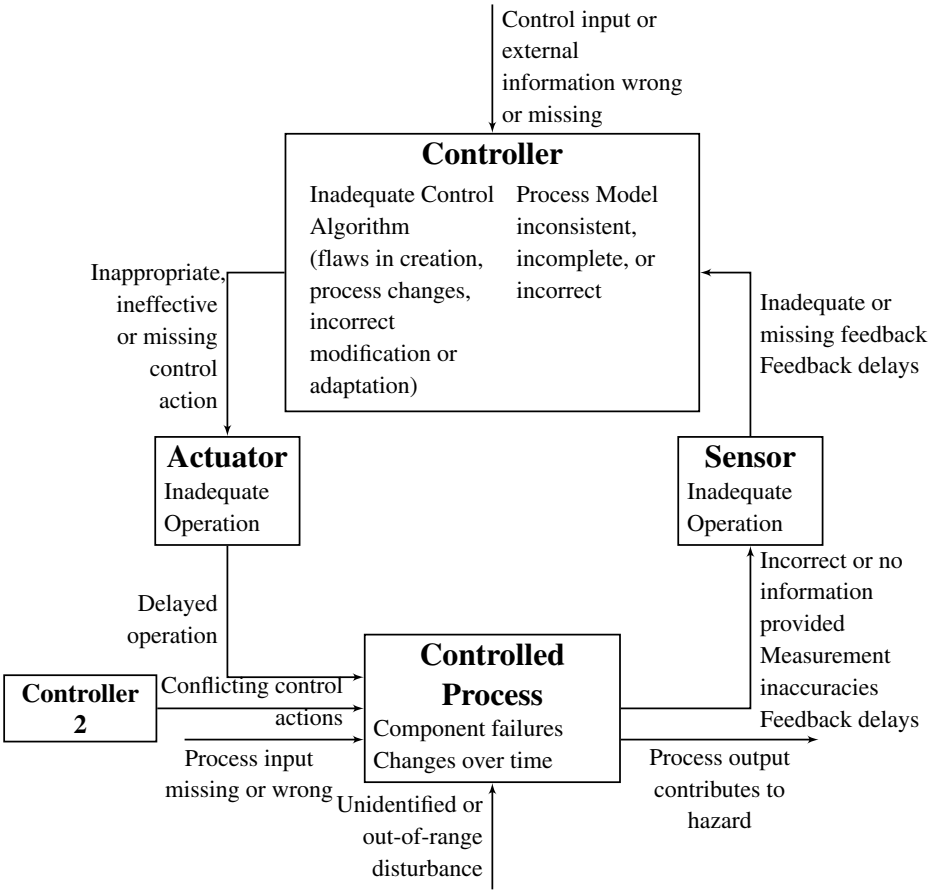



**Fig. 6 STPA causal analysis.**

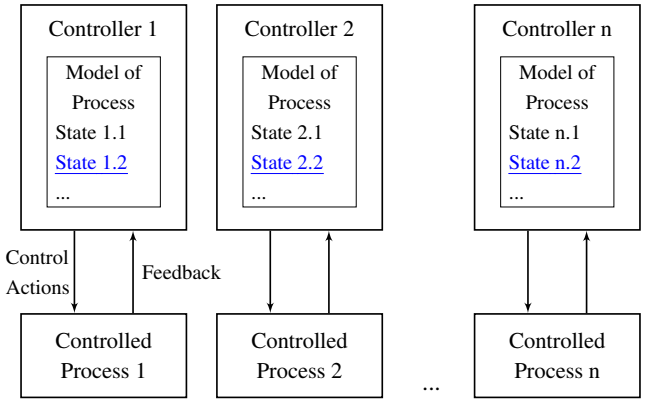



**Fig. 7 Global Process Model states.**

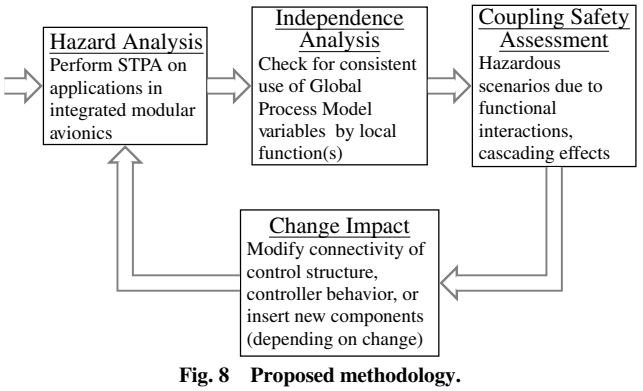



**Fig. 8 Proposed methodology.**

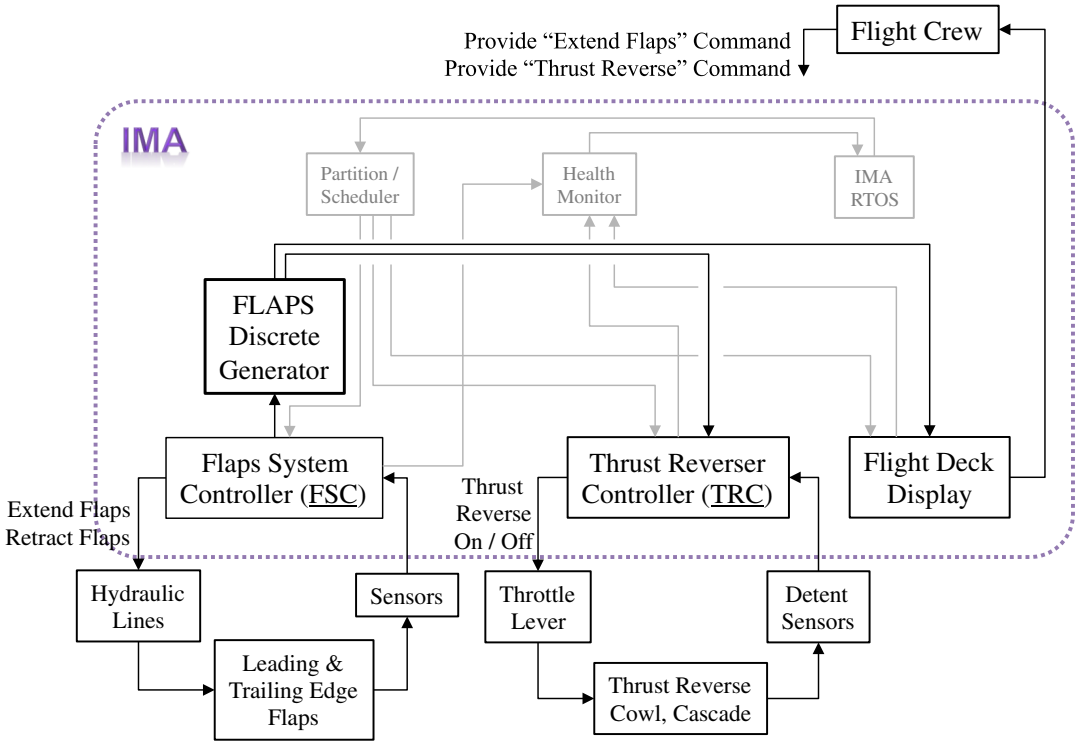



**Fig. 9 Flaps-extended IMA control structure (TR, thrust reverse; LE, leading edge; TE, trailing edge).**

controller is responsible for thrust reverse. The control system diagram includes some of the actuator components used to control these processes as well as generic sensors. The flight deck display is part of a higher-level control loop involving the flight crew, where the display provides feedback for many different process states.

The hazards for any aircraft fall into the general list shown in Table 1 [36]:

For this example, the analysis focuses on [H-2] and [H-4] from Table 1, and these hazards can be refined to reflect the two[§] control systems comprising the example IMA system, flaps, and thrust reverse (see Table 1).

Figure 9 lists some of the control actions (or safety-related responsibilities) of each subsystem, including extend/retract flaps for the flaps system controller and power on/off the thrust reverser for the thrust reverse controller. The flight crew is responsible for providing the "extend flaps" command and "thrust reverse" command.[¶] This analysis does not explicitly consider other vital components of an IMA system such as the partition scheduler, health monitor, and RTOS.

With the completion of the system-level hazards and control structure, the next step (as noted previously) identifies unsafe control actions for each controller in the system according to the four types of unsafe actions in Sec. III.A. Figure 10 shows the unsafe control actions for the flight crew and thrust reverse, where each column represents one of the four types of unsafe control actions. Note that each table highlights a particular unsafe control action; extend flaps "not provided" leads to insufficient lift on takeoff/landing, and thrust reverse "provided" causes bypass air impingement on leading-edge flaps.

STPA step 2 identifies potential causes using the guidewords from control theory in Fig. 6. To obtain a more complete set of causes or scenarios, an analyst should systematically work around the loop, using the guide-words in each box or on each arrow, to identify causes relative to a particular system. However, the following analysis focuses on just the feedback portion of the loop.

Figure 11a depicts a potential cause of the unsafe control action highlighted in Fig. 10: thrust reverse provided when leading-edge flap is extended, causing bypass air impingement. In this case, if the flaps control logic sends the flaps-extended variable if, and only if, the leading-edge flap is in the 1 detent, the thrust reverse control may send an on command when the leading-edge flap is in the path of the thrust reverse exhaust (Fig. 11b).

As another example, Fig. 12 depicts a potential cause of the unsafe control action highlighted in Table 1 for the flight crew: flaps not extended during takeoff or landing. The flight crew's responsibility is only complete when the flap is fully extended. If the algorithm for generating the flaps extension variable uses "flaps not in up detent" (again, see Fig. 11b), then the flight crew may believe their responsibility is completed before it actually has been.

This short example illustrates how the STPA causal analysis can be applied to individual components and controllers for an avionics suite. Decomposing the analysis into individual control loops (and individual unsafe control actions) allows the analyst to systematically identify causes related to all relevant hazards of the system. However, this decomposition makes it difficult to account for coupling that might exist between the behavior of one controlled process and another. The next step accounts for the types of unforeseen coupling that STPA did not explicitly consider previously.

### 2. *Independence Analysis*

The independence analysis procedure first generates a list of Global Process Model states. Any component of the controlled system that is common to multiple controllers must be in the Global Process Model. Once this Global Process Model has been generated, the next step is to check for (in)consistent use of these states, which is typically due to misunderstandings in how the variables are generated, rates at which the variables are updated, or conflicting controller responsibilities.

[§]Note that the flight deck display is not a controller. It is considered as part of the hazard analysis, as will be shown.

[¶]Only those responsibilities associated with thrust reverse and flap position are included here.

**Table 1 General aircraft hazards**

| Category | Description |
|---|---|
| [H-1] | A pair of controlled aircraft violate minimum separation standards. |
| [H-2] | Controlled flight into terrain. |
| [H-2.1] | Loss of lift. |
| [H-3] | Aircraft enters unsafe atmospheric region. |
| [H-4] | Aircraft enters uncontrolled state. |
| [H-4.1] | Loss of lift. |
| [H-4.2] | Structural damage to flaps. |
| [H-5] | Aircraft enters unsafe attitude (excessive turbulence or pitch/roll/yaw that causes passenger injury but not necessarily aircraft loss). |
| [H-6] | Aircraft enters a prohibited area. |



For the flaps control example, identifying Global Process Model variables can be done manually, but for a more complex example, one might employ a search tool on all the controller process models to identify common components. There are three steps in the independence analysis:

1) Identify Global Process Model variable(s).

2) Examine each controller's use of the Global Process Model variables using STPA results.

3) Analyze for potentially inconsistent use of Global Process Model variables (particularly with respect to issues of timing and logic).

Each of these steps will again be illustrated using the flaps controller example.

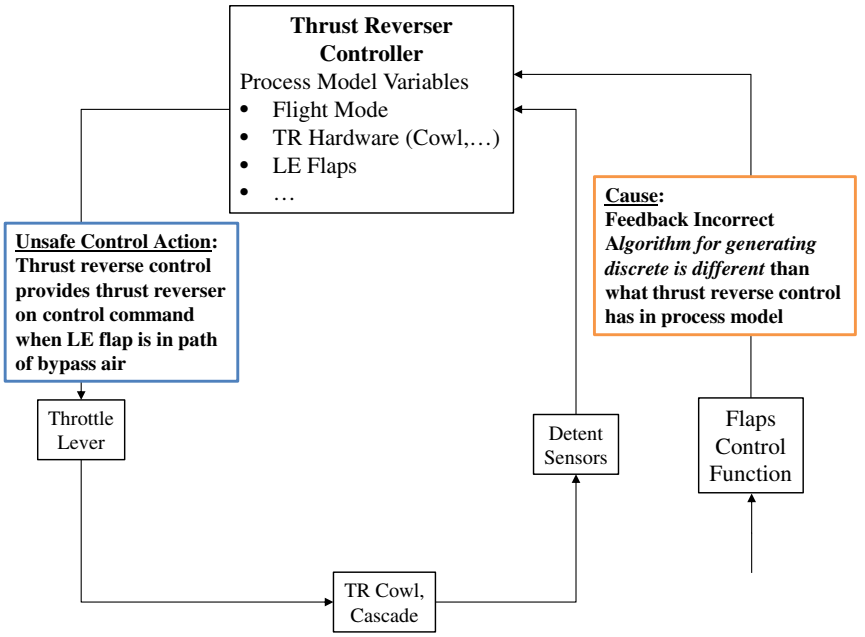
As shown in Fig. 13a, both the thrust reverse controller and flight crew have the "flaps" position in their respective process models. Thus, the flaps state is a Global Process Model variable and is highlighted in the control loops of Fig. 13a. Figure 13b shows how the thrust reverse controller and flight crew (via the flight deck display) may exhibit hazardous behavior due to inconsistent use of the flaps variable. The comparison in Fig. 13b draws from the original hazard analysis performed in the previous subsection. Recall the causal analysis for the thrust reverse controller and flight crew, where the thrust reverse controller and flight crew have hazard causes due to feedback from the flaps system. This section has described only a small subset of the types of inconsistencies the analysis can find. More would be found by considering different types of feedback, increased coupling, and more unsafe control actions.

The inconsistent use of the flaps process model variable is due to ambiguity in the generation of the flaps-extended variable. Recall from the Background section (Sec. II) that valid rules for generating flaps extended include 1) flap surfaces detected in the 1 flap detent, 2) flap surfaces not

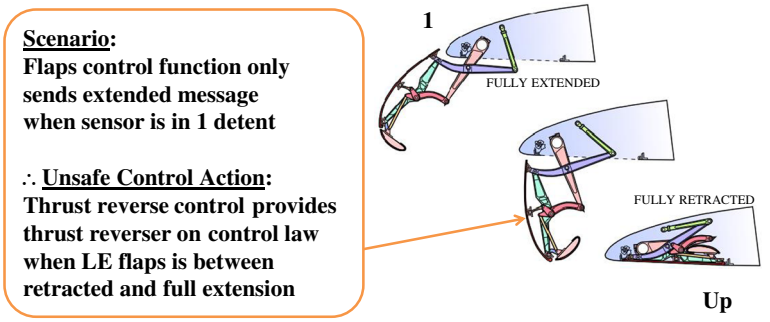
| Controller: Flight Crew | NOT PROVIDED WHEN REQUIRED FOR SAFETY | PROVIDING CAUSES HAZARD | TOO SOON, TOO LATE, OUT OF SEQUENCE | STOPPED TOO SOON, APPLIED TOO LONG |
|---|---|---|---|---|
| Control Action: Extend Flaps | Flaps not extended during takeoff or landing (insufficient lift during terminal ops, climb) | LE flaps extended during thrust reversal (exhaust impingement)<br>Flaps extended during cruise or excessive airspeed & density (flap overload) | Flaps extended too soon during approach (increased drag, loss of speed, flap overload)<br>Flaps extended too late during approach (overspeed, missed runway) | Flaps do not achieve desired angle (e.g. stopped at incorrect discrete) |

| Controller: Thrust Rev Ctl | NOT PROVIDED WHEN REQUIRED FOR SAFETY | PROVIDING CAUSES HAZARD | TOO SOON, TOO LATE, OUT OF SEQUENCE | STOPPED TOO SOON, APPLIED TOO LONG |
|---|---|---|---|---|
| Control Action: Thrust Reverse ON | No thrust reverse on short runway* (runway overshoot)<br>Rollout takes longer than expected (conflict with other taxiing/runway operations) | Reverse thrust during flight leads to loss of velocity, $v$, and therefore lift, $C_L$<br>Bypass air impinges on LE flaps | Reverse thrust applied too soon before landing, resulting in loss of airspeed during approach<br>Applied too late during rollout (Needed when $C_L$ and high $v$ limit effectiveness of friction brakes located on landing gear) | Stopped before aircraft reaches desired speed on runway |

**Fig. 10 Unsafe control actions: flight crew and thrust reverse controller.**




**a) STPA causal analysis (feedback only)**




**b) Flaps extended scenario (flap graphic adapted from [32])**

**Fig. 11 Thrust reverse causal analysis [37].**

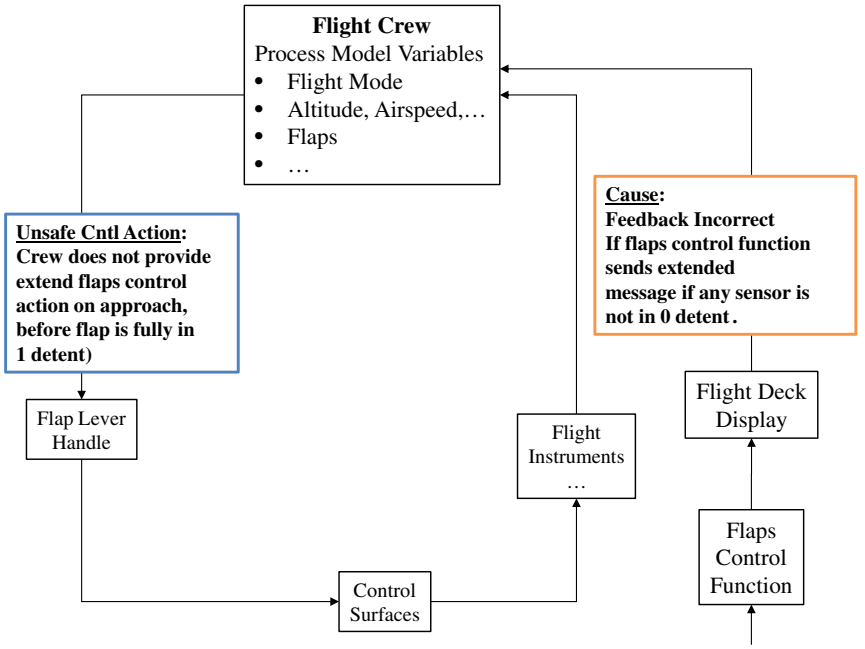
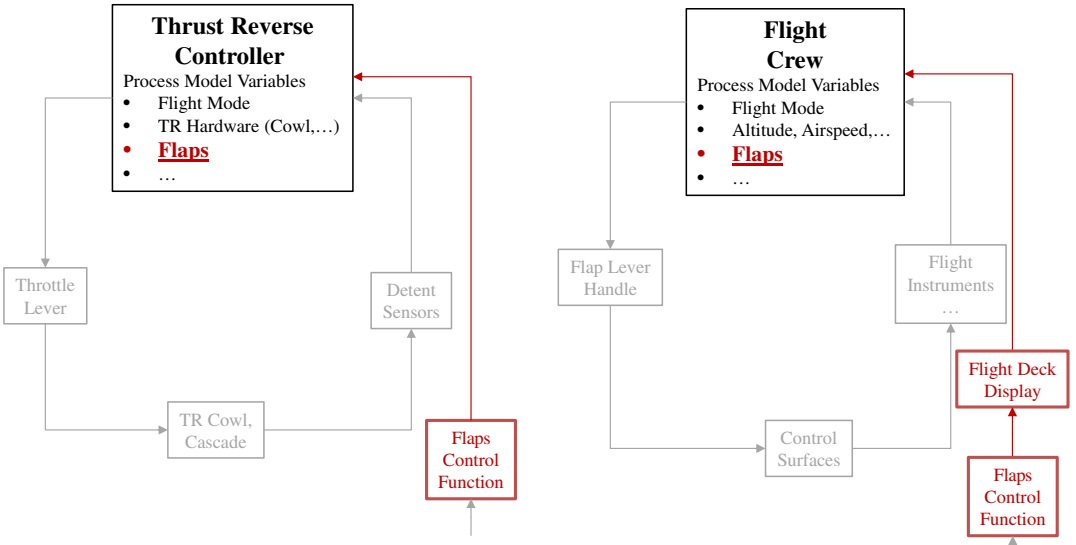



**Fig. 12 Flight crew causal analysis (feedback only).**




**a) Global process model identification**

| Thrust Reverse Controller: | Flight Deck Display: |
|---|---|
| Needs flaps extended variable whenever flap surface is not in 0 detent | Needs flaps extended variable only when flap surface is in 1 or greater detent |
| Assumptions: | Assumptions: |
| Thrust reverse control risks impingement on flaps any time LE flaps are not stowed | Crew responsibility only complete when flaps make it fully to detent |

**b) Inconsistent use of GPMV**

**Fig. 13 Independence analysis.**

detected in the up flap detent, or 3–4) similar rules measuring the flap lever handle instead of the flap detent. Note here that, even for this trivial example, there is a subtle result. The thrust reverse controller and flight deck display have no data interface, and without further analysis, one could assume that they behave independently. The behavior of the thrust reverse controller and flight deck display are indeed not directly dependent on one another. However, what this analysis has shown is that their behavior is not independent, and their potential for hazardous behavior is indirectly coupled through the use of a Global Process Model component. The indirect coupling between the thrust reverse controller and flight deck display is an example of emergence that cannot be identified by analyzing the individual software applications and their use of input variables. STPA, supplemented with the Global Process Model variable, allows the analyst to identify these kinds of emergent properties.

### 3. *Coupling Safety Assessment*

The primary objective of the third type of analysis, coupling safety assessment, is to generate additional system constraints based on the results of the hazard analysis and independence analysis. In particular, the coupling safety assessment identifies strategies that eliminate or mitigate inconsistent usage of the Global Process Model. The previous two subsections identified potential sources of hazardous behavior of the example IMA system due to inconsistent use of the flaps Global Process Model variable. The inconsistent use of the flaps variable is due to the ambiguity in the definition of how this variable is generated. The flight crew and the thrust reverse controller have different safety-related responsibilities during takeoff and landing, and therefore have different expectations of what flaps extended should mean physically.

Instead of the four definitions of flaps extended, shown in Sec. II.D and represented in Algorithm 1, the analyses in the previous two subsections suggest several alternatives for the flaps-extended generation logic. The simplest solution in this example is to choose one of the four valid conditions shown in Algorithm 1. Algorithm 2 states simply that the flaps-extended variable should be generated if, and only if, the flap surfaces are detected in the 1 or greater detents. It might be fairly obvious by inspection that such a trivial solution will eliminate one of the causes generated in the previous subsection but not the other cause. However, the goal of this paper is to illustrate a systematic method for identifying behavioral

**Algorithm 1 Original (ambiguous) flaps-extended logic (see Sec. II.D)**

**if** Both left and right trailing-edge flap surfaces detected in the 1 or greater flap detent, **then**
  Flaps Position→EXTENDED
**else if** Both left and right trailing-edge flap surfaces detected not in the Up flap detent, **then**
  Flaps Position→EXTENDED
**else if** Flap Lever Handle detected in the 1 or greater flap handle detent, **then**
  Flaps Position→EXTENDED
**else if** Flap Lever Handle detected not in the Up flap handle detent, **then**
  Flaps Position→EXTENDED
**end if**

**Algorithm 2 Modified flaps-extended logic**

**if** Both left and right trailing-edge flap surfaces detected in the 1 or greater flap detent, **then**
  Flaps Position→EXTENDED
**else**
  Flaps Position→EXTENDED
**end if**

flaws that may lead to hazardous behavior, using these results to generate constraints and modifications to the behavior, and then checking if the modifications resolve any issues or create new types of hazardous behavior.

Note also that there are many other types of changes that could resolve inconsistent uses of a Global Process Model variable, including structural modification, component behavior modification, or changes in information exchange between components. The previous paragraph suggests modifying the algorithm used to generated the flaps-extended variable. In terms of the control structure shown in Fig. 9, this is a component change. That is, the internal behavior of a component of the control structure has been modified, and in this case, the source of the Global Process Model variable is modified. Alternatively, other components' behaviors may be modified. For example, the functions that use the Global Process Model variable could be modified, in this case, to account for timing differences between when a flap begins extension and reaches its detent. Human behavior cannot simply be modified in the same sense as software, but airframe manufacturers or regulators could also add procedures for the flight crew to account for inconsistent usage of Global Process Model variables. Procedures and training are not necessarily the most elegant solution but certainly could be considered as part of a thorough design study.

Additionally, the designers may decide to add a dedicated flaps variable generator for each user function.[**] This fundamentally changes the system's control structure, and such a change is called a "structural modification." Finally, the behavior of the connectors (the lines with arrows) in the control structure in Fig. 9 might be modified, which does not affect the connectivity of the components nor the components' behavior. This is an "information exchange modification," and available types of changes include update rate, communication protocol, message structure and sequence, precision, and so on.

To reiterate, the objective of the coupling safety assessment is to use both the hazard analysis and independence analysis to generate modifications that can resolve conflicts among IMA functions and other actors in the system. The independence analysis identifies which hazard causes need to be resolved, but the STPA hazard analysis is also instructive in terms of elucidating possible design alternatives. STPA identifies potential flaws in how the system is controlled, and the previous paragraphs briefly suggested how the control structure may be used to identify design alternatives or modifications. These modifications include changes in component behavior (including software logic and/or procedural elements for human operators), changes in connectivity of the control structure itself, and changes in the way that information is exchanged between components. Once design modifications have been explored and selected, the next step involves the assessment of whether and how these changes have impacted the existing hazard analysis.

#### 4. *Change Impact Hazard Analysis*

Change impact hazard analysis involves analyzing modifications of the system that (hopefully) eliminate the identified hazardous scenarios. One aspect of the change impact analysis is to identify assumptions in the original STPA hazard analysis that are no longer valid. Examples include modifying control structure connections; modifying controller behavior; or inserting new components such as actuators, feedback, or entirely new controllers. Any of these changes could result in a new set of unsafe control actions or different types of causes (see Sec. III.A and Fig. 6). For example, changing the timing requirement of a computed variable might result in feedback delays, as in the upper right arrow of Fig. 6.

Hazard analyses require a significant level of effort and put resource burdens on both system developers and regulators. Section II explored some of the potential benefits of IMA, including the notion that modular software applications may minimize the impact of change. However, Sec. II.D described some limitations of this approach and suggested that more and different kinds of analyses are necessary to ensure that modular applications behave safely upon integration. Despite these limitations, the move to minimize the impact of change is still a valid and desirable goal, particularly in hazard analysis. This section outlines a few brief ideas about how to minimize the burden of reanalysis whenever system behavior is modified. In the context of this paper, system modifications are the result of inconsistencies found in the independence analysis, but system changes clearly come from other sources. For example, the airframe manufacturer may simply decide to add a new application to provide additional features for customer satisfaction, or application software may be replaced due to changes in hardware or system design. This paper explicitly focuses on changes due to the coupling safety assessment, but the ideas provided in the following paragraphs should be applicable to all types of change.

Safety depends on the accuracy of the assumptions and models underlying the design and hazard analysis processes [23]. It is therefore the duty of the analyst and designer to ensure that environmental and system assumptions are enforced in the design and, conversely, to find invalid assumptions. The fundamental aspect of change impact hazard analysis is to identify those assumptions in the original hazard analysis that have changed. (Of course, the preceding statement has its own assumption; that the original analysis is thorough and accurate.)

Figure 14 depicts how an assumption in the existing hazard analysis can change (or not). Because of the ambiguity underlying the flaps-extended variable, the original analysis assumed that the flaps controller could use any of the criteria to generate the discrete.[††] Clearly, the modification in logic in Algorithm 2 eliminates this assumption, and now the analyst must decipher if this change is positive, negative, or neutral with respect to hazardous behavior. In Fig. 14, the modified assumption has actually eliminated a particular cause of hazardous behavior for the flight crew. The crew's responsibility is only completed when the flap is fully extended, and the modification in logic eliminates the particular cause of feedback due to the flight display. The modified logic has not, however, changed the causation for the thrust reverse controller, where the thrust reverse controller does not shut of the thrust reverse until the flap is all the way in the 1 detent.

Clearly, more modifications are necessary. The flaps-extended example would require the introduction of more variables to represent the state of the flaps, and perhaps further modification of both the flaps generation logic and the logic of user functions. The flaps-extended example is trivial, but it is illustrative of the problems associated with assuming that IMA functions with no direct data interface actually behave independently. The procedure described in this paper provides a systematic way to identify potential modifications, ascertain how a modification affects the existing hazard analysis, and minimize workload by avoiding rework for aspects of the system for which the assumptions have not changed.

**Adding dedicated functions would probably no longer be considered IMA and are more aligned with a federated approach.

[††]See Sec. II.D and Algorithm 1 for details on this ambiguity, but it is due to that fact that flaps extension can either mean "not fully stowed" or "fully extended" and can be measured directly at the flap or at the control lever.

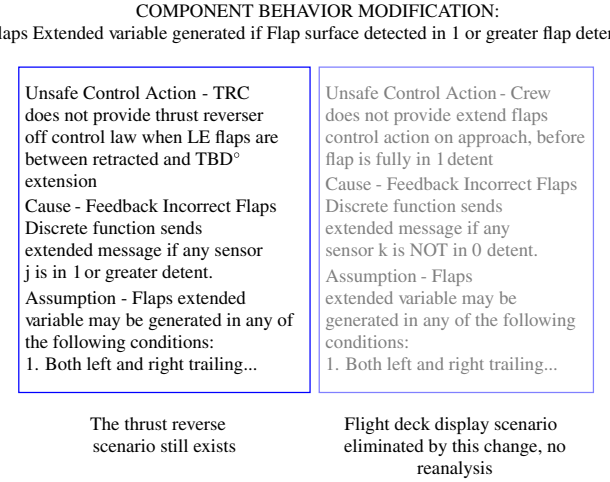



**Fig. 14 Change impact hazard analysis.**

## IV. Conclusions

This paper introduces a systematic method for identifying hazardous behavior due to interaction among software applications on an IMA platform. The method uses STPA, which frames safety as a control problem rather than a reliability problem, to identify hazardous scenarios due to software behavior. An important aspect of enforcing safe behavior is the maintenance of an accurate model of the controlled process, particularly for highly coupled, software-intensive systems. Thus, the method introduces the concept of a Global Process Model. The analysis checks for inconsistent Global Process Model states among software applications that have different safety-related responsibilities.

The paper then suggests a strategy for eliminating or mitigating against inconsistent Global Process Models. Finally, the paper presents a method for analyzing the impact that the elimination/mitigation strategies have on the original hazard analysis.

The procedures in this paper are illustrated using a simple IMA system with three basic aircraft functions. One important question about the viability of this method pertains to scalability. It is encouraging that STAMP and STPA, which provide the theoretical and analytical framework for the methods presented here, have been used to analyze very large, complex systems, both in aerospace and other domains [30–32]. The demonstration in this paper was performed manually, which could be a significant limitation when analyzing larger, more complex systems. However, Thomas [33] and Thomas and Leveson [34] have extended STPA to automate certain aspects of the analysis. This work has the potential to be integrated into the methods described here, and automated tools that identify Global Process Model Variables could also be developed.

STPA identifies certain types of causal factors that are not considered when using failure-based hazard analysis techniques. The flaps control example in Sec. III shows how the method identifies potential inconsistencies in data usage that can lead to hazardous behavior. These inconsistencies cannot be found by analyzing application software individually because they only occur when multiple software applications have access to shared variables. The method proposed in this paper also has the potential to reduce workload because it identifies only those variables that have the potential to cause hazardous behavior.

## Acknowledgments

This research was partially supported by NASA under research contract NNL10AA13C. We would like to thank Michael Holloway at NASA for his assistance and support.

## References

[1] "Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations," Radio Technical Commission for Aeronautics DO-297, Washington, D.C., 2005.
[2] Baker, K., "Filling the FAA Guidance and Policy Gap for Systems Integration and Safety Assurance," *30th Digital Avionics Systems Conference*, IEEE, Piscataway, NJ, 2011, pp. 1B4-1–1B4-4.
[3] Bartley, G., and Lingberg, B., "Certification Concerns of Integrated Modular Avionics (IMA) Systems," *27th Digital Avionics Systems Conference*, IEEE, Piscataway, NJ, 2008, pp. 1.E.1-1–1.E.1-12.
[4] Lewis, J., and Rierson, L., "Certification Concerns with Integrated Modular Avionics (IMA) Projects," *22nd Digital Avionics Systems Conference*, IEEE, Piscataway, NJ, 2003, pp. 1.A.3–1.1-9.
[5] "Certication Considerations for Highly-Integrated or Complex Aircraft Systems, Revision A," Society of Automotive Engineers, SAE International, SAE-ARP4754A, 2010.
[6] "Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment," Society of Automotive Engineers, SAE International, SAE-ARP4761, 1996.
[7] Watkins, C. B., and Walter, R., "Transitioning from Federated Avionics Architectures to Integrated Modular Avionics," *26th Digital Avionics Systems Conference*, Vol. 1, IEEE, Piscataway, NJ, 2007, pp. 2.A.1-1–2.A.1-10.
[8] Watkins, C., "Integrated Modular Avionics: Managing the Allocation of Shared Intersystem Resources," *25th Digital Avionics Systems Conference*, IEEE, Piscataway, NJ, 2006, pp. 1–12.
[9] Ramsey, J. W., "Integrated Modular Avionics: Less is More-Fresh Approaches to Integrated Modular Avionic Architectures will Save Weight, Improve Reliability of A380 and B787 Systems," *Avionics Magazine*, Vol. 31, No. 2, 2007, p. 24.
[10] Vincoli, J. W., "Basic Guide to System Safety," Wiley-Interscience, New York, 2006, p. 201.
[11] Wang, S., and Liu, Y., "A Survey of System Safety Technique of Commercial Aircraft," *9th International Conference on Reliability, Maintainability and Safety (ICRMS)*, IEEE, Piscataway, NJ, 2011, pp. 504–512.
[12] Rushby, J., "New Challenges in Certification for Aircraft Software," *Proceedings of the Ninth ACM International Conference on Embedded Software*, ACM, New York, 2011, pp. 211–218.
[13] Graydon, P., and Kelly, T., "Assessing Software Interference Management When Modifying Safety-Related Software," *SAFECOMP*, Springer, Berlin, 2012, pp. 132–145.

[14] Prisaznuk, P., "Arinc 653 Role in Integrated Modular Avionics (IMA)," *27th Digital Avionics Systems Conference*, IEEE, Piscataway, NJ, 2008, pp. 1.E.5-1–1.E.5-10.
[15] Lalli, V. R., Kastner, R. E., and Hartt, H. N., "Training Manual for Elements of Interface Definition and Control," NASA TR-1370, 1997.
[16] Espinoza, H., Ruiz, A., Sabetzadeh, M., and Panaroni, P., "Challenges for an Open and Evolutionary Approach to Safety Assurance and Certification of Safety-Critical Systems," *IEEE First International Workshop on Software Certification (WoSoCER)*, IEEE, Piscataway, NJ, 2011, pp. 1–6.
[17] Page-Jones, M., "*The Practical Guide to Structure Systems Design*," Prentice–Hall, Upper Saddle River, NJ, 1988, pp. 63–66, 69–73.
[18] "Boeing B747-400 G-BYGA Group 'A' L/E Flaps Retracted on Takeoff from O.R. Tambo Airport, South Africa," South African Civil Aviation Authority Final Rept. CA18/3/2/0717, Halfway House, South Africa, June 2010.
[19] *Systems Engineering Handbook*, NASA, 2007, p. 138.
[20] Conmy, P., and McDermid, J., "High Level Failure Analysis for Integrated Modular Avionics," *6th Australian Workshop on Safety Critical Systems and Software*, Vol. 3, ACM, New York, 2001, pp. 21–31.
[21] Conmy, P., Nicholson, M., and McDermid, J., "Safety Assurance Contracts for Integrated Modular Avionics," *Proceedings of the Eighth Australian Workshop on Safety Critical Systems and Software*, ACM, New York, 2003, pp. 69–78.
[22] Checkland, P., *Systems Thinking, Systems Practice*, Wiley, New York, 1981, pp. 74–82.
[23] Leveson, N. G., *Engineering a Safer World*, MIT Press, Cambridge, MA, 2012, pp. 63–67.
[24] Yong, C., Zexin, W., Xupo, O., and Liang, Z., "Approach Civil Integrated Modular Avionics Airworthiness Certification by Iterative Incremental Certification Process," *IEEE 2nd International Conference on Digital Object Identifier*, IEEE, Piscataway, NJ, 2010, pp. 148–151.
[25] Ruiz, A., Habli, I., and Espinoza, H., "Towards a Case-Based Reasoning Approach for Safety Assurance Reuse," *Computer Safety, Reliability, and Security*, Vol. 7613, Lecture Notes in Computer Science, 2012, pp. 22–35.
[26] Kletz, T. A., *HAZOP and HAZAN: Identifying and Assessing Process Industry Hazards*, Inst. of Chemical Engineersm Rugby, England, U.K., 1992, pp. 9–20.
[27] Gould, J., Glossop, M., and Ioannides, A., "Review of Hazard Identification Techniques," Health and Safety Lab. Rept. HSL/2005/58, Sheffield, England, U.K., 2005.
[28] Kletz, T., *Computer Control and Human Error*, Gulf Professional Publ., New York, 1995, pp. 45–56.
[29] Hulin, B., and Tschachtli, R., "Identifying Software Hazards with a Modified Chazop," *PESARO 2011: The First International Conference on Performance, Safety and Robustness in Complex Systems and Applications*, XPS (Expert Publishing Systems), Wilmington, DE, 2011, pp. 7–12.
[30] Ishimatsu, T., Leveson, N. G., Thomas, J. P., Fleming, C. H., Katahira, M., Miyamotoand, Y., Ujiie, R., Nakao, H., and Hoshino, N., "Hazard Analysis of Complex Spacecraft Using Systems-Theoretic Process Analysis," *Journal of Spacecraft and Rockets*, Vol. 51, No. 2, 2014, pp. 509–522. doi:10.2514/1.A32449
[31] Leveson, N., Couturier, M., Thomas, J., Dierks, M., Wierz, D., Psaty, B. M., and Finkelstein, S., "Applying System Engineering to Pharmaceutical Safety," *Journal of Healthcare Engineering*, Vol. 3, No. 3, 2012, pp. 391–414. doi:10.1260/2040-2295.3.3.391
[32] Pereira, S. J., Lee, G., and Howard, J., "A System-Theoretic Hazard Analysis Methodology for a Non-Advocate Safety Assessment of the Ballistic Missile Defense System," Defense Technical Information Center (DTIC) Document, ADA466864, Fort Belvoir, VA, 2006.
[33] Thomas, J., "Extending and Automating a Systems-Theoretic Hazard Analysis for Requirements Generation and Analysis," Sandia National Labs. Rept. SAND2012-4080, Albuquerque, NM, 2012.
[34] Thomas, J., and Leveson, N., "Performing Hazard Analysis on Complex, Software- and Human-Intensive Systems," *29th International System Safety Conference*, International System Safety Society, Unionville, VA, 2011.
[35] Suo, D., An, J., Wu, J., and Zhu, J., "Filling the Gap Between IMA Development and Safety Assessment Through Safety-Driven Model-Based System Engineering," *31st Digital Avionics Systems Conference*, IEEE, Piscataway, NJ, 2012, pp. 1–22.
[36] Fleming, C. H., Spencera, M., Thomasa, J., Levesona, N., and Wilkinson, C., "Safety Assurance in Nextgen and Complex Transportation Systems," Vol. 55, *Safety Science*, 2013, pp. 173–187.
[37] Bislins, W., "Variable Camber Leading Edge Flaps (Cross-Section)," US Patent US4262868A.

M. Davies
*Associate Editor*