

# FAILURE ANALYSIS AND THE SAFETY-CASE LIFECYCLE

William S. Greenwell, Elisabeth A. Strunk, and John C. Knight  
*Department of Computer Science, University of Virginia*

**Abstract:** The failure of a safety-critical system, though undesirable, is often a source of valuable lessons that can help prevent future failures. Current analysis practices do not always yield as much knowledge as they might about possible flaws in the system safety argument. In this paper, we introduce the lifecycle for safety cases. We use it to develop a framework to guide the analysis process and the development of lessons and recommendations. We illustrate the ideas with an example using the failure history of an air-traffic-control safety system.

**Key words:** failure analysis, safety cases, assurance

## 1. INTRODUCTION

Safety-critical systems are engineered to prevent failure, but, despite this, accidents and incidents sometimes occur. Developers create safety arguments for each safety-critical system they produce, even though the arguments might be informal, flawed, or undocumented. A safety-related failure of the system implies that there is a flaw in the safety argument, and continuing to operate the system without reassessing the safety argument—even if the immediate cause of the problem has been identified and corrected—is potentially dangerous.

In this paper, we use the safety case as a framework to embody a safety argument, showing how that argument can guide failure analysis and how failure analysis can be used to update the argument. Although safety-case maintenance has been studied, current theory does not combine safety-case maintenance with current work in accident investigation. We introduce a

comprehensive, post-deployment lifecycle for the safety case in which detailed and explicit feedback paths help maximize the benefits realized from any failure. The lifecycle updates the safety case throughout the system's life so that the safety case continues to provide a convincing argument that the system is safe, even after a failure.

The paper is organized as follows. Section 2 describes the safety case, introducing it as the focus not only of system assurance but also of failure treatment. Section 3 details the specifics of our lifecycle framework, and Section 4 gives an example of how it might be applied. Section 5 then concludes the work.

## **2. SAFETY CASES**

In safety-critical systems, the primary loss to be addressed is undesired system behavior that will lead to death or injury to persons or damage to property. Such loss could be extremely serious; in most cases, the system developers are regulated by an external body to ensure that the potential for loss has been adequately assessed and addressed. Assuring system safety, however, is a formidable task. A system cannot be proven to be safe unless it has been operated over all possible inputs to ensure that it can never reach a hazardous state. Any other method of assurance, however well-formed, leaves open the possibility that the assured system properties are insufficient or invalid. For most systems, however, exhaustive testing is impractical, and for others it is impossible.

To avoid this problem, developers rely on other assurance techniques that produce arguments, but not proofs, that the probability of a system's failing in a way that could potentially cause a violation of safety properties is below a maximum acceptable threshold. Several widely-used industry standards for arguing assurance of safety rely on a process-based approach, such as RTCA DO-178B (Weaver, 2003; RTCA, 1992). These approaches assume that following a prescribed set of development processes will result in the production of a safe system. There is, in fact, little evidence to support this assumption (McDermid, 2001), and in process-based safety arguments, there is no way to comprehensively analyze the argument if a system failure occurs. Lessons could only be applied to the process, and there might not be an obvious causal relationship from a particular part of the process to a particular system failure.

Hamilton and Rees (1999) and Weaver (2003) argue that, instead of prescribing a particular process, standards should instead specify what types of evidence developers can generate to show that their systems meet the necessary safety requirements. An example of this type of assurance

structure is the safety case, which argues that a system is safe to use in its intended environment. More specifically, it argues that the risks associated with the operation of the system have been reduced to an acceptable level.

To look at the various elements that make up a safety case, we summarize the *Goal Structuring Notation (GSN)*, a graphical notation developed at the University of York for depicting safety arguments (Kelly, 1998). The elements of an argument modeled in GSN are shown in Fig. 1.

In the rest of this paper, we use the elements of GSN as a model for the different pieces of information a safety case should contain because GSN was created precisely for the purpose of assisting engineers in making structured safety arguments.

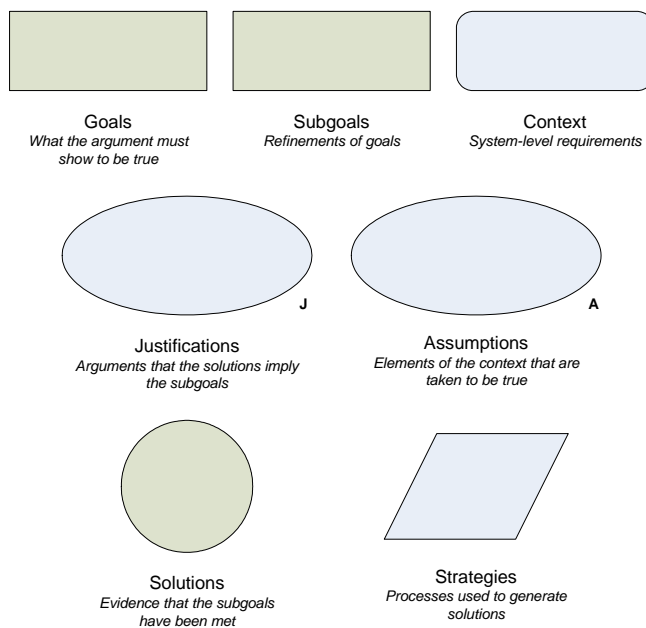


Figure 1. Basic Elements of GSN (Kelly, 1998)

### 3. THE ENHANCED SAFETY-CASE LIFECYCLE

An important feedback path exists between system development and failure analysis. Faults often manifest themselves after deployment, and the role of analysis after a failure is to uncover these faults. Typically, once a failure—either an incident or an accident—has been analyzed, the lessons

and recommendations from the analysis are fed back into the development lifecycle to help prevent future failures.

Ideally, this feedback mechanism would ensure that all the benefits of failure analysis would be realized. However, the complexity of modern safety-critical systems makes the systems very difficult to analyze. Coupled with the informality of the failure analysis process, this complexity reduces the level of confidence that the lessons obtained from an analysis are comprehensive and correct. In addition, differences in designs and in development practices can limit the scope of lessons and recommendations significantly. Opportunities to discover and correct faults in systems and flawed development practices are sometimes missed; we discuss an example of this problem in Section 4.

Safety cases employing evidence-based assurance offer a solution to this problem by providing a rigorous basis for the feedback path between system development and failure analysis. The core safety-case lifecycle has been introduced by Kelly and McDermid; it is the process through which updates are made to the safety case for a given system when a challenge arises (Bishop, 1998; McDermid, 1999; Kelly, 1998). One of the challenges they note is that of a mishap (an accident or incident). We introduce the enhanced safety-case lifecycle, which uses the safety case to guide failure analysis after a mishap and to update the safety case based on the results of the analysis.

A failure is evidence that a system's original safety argument was flawed; the system was in fact less safe than expected. Thus, for any particular failure, we define *two* safety cases: the pre-failure safety case and the post-failure safety case. The former is the original safety case that was developed for the system before the failure. The post-failure safety case is the pre-failure safety case amended to reflect the results of the failure analysis. The post-failure safety case is essentially the result of correcting the original, flawed safety argument.

The fact that the original safety case was flawed and had to be updated as a result of the analysis of a failure leads to the enhanced safety-case lifecycle (illustrated in Fig. 2). The safety case is subject to revision over time based on experience, and careful and systematic determination of the essential changes provides the basis for both the analysis process and the determination of lessons and recommendations.

In order to elaborate the framework and add rigor to the feedback path, it is necessary to note that all of the basic elements of a safety case—including assumptions and contextual information—are first-class objects within the rigorous structure of the safety case. The failure analysis process that we have developed is based on examining these objects in context in the safety case and determining which are somehow defective following an accident.

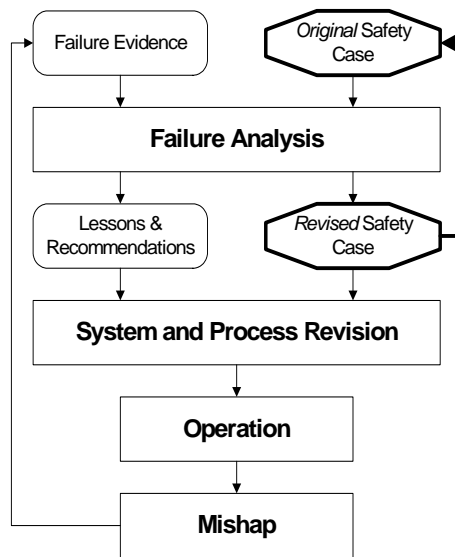


Figure 2. The Enhanced Safety-Case Lifecycle

Viewing each element as a first-class object ensures that the analysis is systematic and covers all aspects of the argument. This contrasts with the traditional approach where assumptions, for example, are usually treated in an ad hoc manner. Treating all of the elements of the safety case in a systematic way is facilitated by basing the analysis on a rigorous safety case.

The essence of our overall approach then is to begin with the original safety case for the system and organize the failure analysis around the process of correcting the safety case. Lessons show the elements of the pre-failure safety case that cannot be used in the post-failure safety case, and recommendations suggest ways to address the lessons.

### 3.1 Safety Goals of Failure Analysis

Investigating failures presents opportunities to learn lessons and prevent future failures. In many cases, however, important lessons are not recognized or essential corrective actions are not taken—such as in the case of the Korean Air flight 801 accident (NTSB, 2001). This stems in part from the lack of a coherent and complete approach to the treatment of failure. Researchers have developed a variety of approaches to the determination of causal factors (Johnson, 2003), but there is no rigorous framework within which lessons are developed, corrective actions defined, corrective actions implemented, and so on.

Accident investigations generally produce two major classes of results: *findings*, an investigation's analysis of the factors contributing to the accident; and *recommendations*, the investigation's suggestions as to how similar accidents could be prevented. The term *lesson* is often applied to findings that expose a flaw in the system or its development process, and recommendations tend to be directed towards fixing the flaws exposed by lessons, whether those lessons are stated explicitly or inferred from the findings. While the meanings of the terms are fairly clear in this informal context, we can define these terms more rigorously using the safety case as the artifact to which they apply.

### 3.2 Lessons Learned About the Safety Case

In the enhanced safety-case lifecycle, the first step in constructing the post-failure safety case is to find the flaws in the pre-failure safety case, i.e., determine the lessons of the incident. The safety case structure can aid the failure analysis process for determining lessons. Two specific techniques for doing this are *backtracking* and *dependency analysis*, such as in Kelly's process of using GSN to support safety-case maintenance (Kelly, 1998). Investigators would begin a failure analysis by assuming that the top-level goal of the safety argument was not satisfied and then backtracking through the argument to isolate faulty assumptions and evidence. Upon identifying a flaw in the safety argument, investigators could then use dependency analysis to determine which parts of the argument are affected by the flaw. Dependency analysis is a powerful tool for assessing the severity of a flaw, particularly if the flaw related to an assumption or piece of evidence used in safety arguments for other systems.

Once investigators have analyzed a failure and identified the flaws in the safety argument that contributed to it, they can prepare the lessons resulting from the analysis. These lessons state the elements of the safety argument that are faulty and in need of revision. For example, a lesson that might be learned from the failure of an advisory system could read:

*“The assumption that a visual alert alone is sufficient to notify the controller of an altitude violation is invalid because the controller might not be at the workstation when the alert is raised.”*

This lesson might derive in the enhanced safety-case lifecycle from a safety case element that relied on the assumption:

*“Any visual alert would be seen by the controller.”*

### 3.3 Recommendations Derived from the Failure Analysis

Recommendations in the enhanced safety-case lifecycle are suggestions made by an investigating team that are intended to help system engineers in the task of creating a valid post-failure safety case. Recommendations can take two forms. The first is that of a revised piece of the safety argument. System engineers would be expected to construct a corresponding revised system design that would allow the use of the argument fragment in the system's updated safety case. For the example lesson given above, a corresponding recommendation might state:

*“It may be assumed that visual and aural alerts are sufficient to notify the controller of an altitude violation, because, although controllers might not be at their workstations when an alert is raised, at least one controller will be in the tower at any given time.”*

In order to use this assumption in a safety argument, a developer would now have to show that the system generates both visual and aural alerts to conclude that the controller will be notified of an altitude violation, which might require changes to the system's design.

The second form a recommendation might take is that of a possible system change, always accompanied by a corresponding postulated change to the pre-failure safety case. If the system engineer chooses to implement recommended system changes, he can use the postulated change to guide the development of the actual post-failure safety case.

Recommendations are weaker statements than lessons. By invalidating an assumption, the effect of a lesson on the system development / failure analysis feedback loop is to forbid the use of that assumption in current and future safety arguments. On the other hand, a recommendation merely offers a potential system change and / or alternative safety argument to use in the updated safety case. Compliance with a recommendation does not automatically imply compliance with a safety goal. While the recommendations can include a potential safety argument, the safety case is a model of system operation and cannot be evaluated apart from the actual system design.

## 4. EXAMPLE: THE MINIMUM SAFE ALTITUDE WARNING (MSAW) SYSTEM

To illustrate the use of the enhanced safety-case lifecycle, we present a hypothetical example based on the Federal Aviation Administration's (FAA) Minimum Safe Altitude Warning (MSAW) system. We selected the MSAW

system for this example because it was conceived, implemented, and has since been revised through an informal interaction between the FAA and the National Transportation Safety Board (NTSB), a process intended to serve a purpose related to that of the enhanced safety-case lifecycle. We step through the development and revision history of MSAW and show how it could be accomplished in a more systematic, rigorous manner.

The MSAW system did not have an explicit safety case defined. By putting it into service, however, the FAA has made an implicit statement that it enhances the safety of the national airspace system. We have attempted to assemble the FAA's implicit safety argument from available system descriptions, accident reports, and communications with FAA officials.

## **4.1 System Description**

MSAW is a software system designed to alert air traffic controllers to aircraft flying below a predetermined minimum safe altitude. The system receives altitude data for tracked aircraft from radar returns and compares the data against a terrain database containing elevations for the surrounding area. If an aircraft's reported altitude is below, or is predicted to descend below, the minimum safe altitude for the region in which it is operating, the system issues an alert to the controller. Upon receiving the alert, the controller is responsible for contacting the aircraft and notifying the flight crew of the hazard.

## **4.2 MSAW Chronology**

Prior to the development of MSAW, the implicit safety argument that controllers would detect and notify low-flying aircraft relied on the assumption that they would be able to manually spot such aircraft on their radar displays by examining the altitudes reported in each aircraft's data block. The radar displays did not provide terrain information, so controllers would also have to know the minimum safe altitude for the region in which each aircraft was flying. In response to a December 1972 accident in which a commercial aircraft crashed near Miami, Florida, the NTSB issued a safety recommendation asking the FAA to "review the ARTS III [air traffic management] program for the possible development of procedures to aid flight crews when marked deviations in altitude are noticed by an air traffic controller" (NTSB, 2001). The FAA responded to this recommendation by implementing the MSAW system in 1977. Despite the deployment of MSAW, accidents that it was designed to prevent have persisted, and the system has been changed numerous times to address them. These accidents are highlighted below.

On September 8, 1989, a commercial aircraft struck four transmission lines while executing an approach to Kansas City International Airport in Missouri. The NTSB found that, although the airport's tower was equipped with MSAW, the system failed to raise an alert because the altitude violation occurred in a region that had been excluded from MSAW processing due to a configuration error. The Board recommended that the FAA provide guidelines to its facilities for configuring the MSAW system to prevent such errors from recurring. The FAA implemented this recommendation in 1993.

On June 18, 1994, a Learjet crashed just short of the runway at Dulles International Airport. Upon investigating the MSAW system at Dulles, the NTSB again found an instance in which the system failed to generate any alerts because it had been configured improperly and recommended that the FAA conduct a "complete national review of all environments using MSAW systems" (NTSB, 2001). The FAA completed this review in 1996.

On January 29, 1995, during the period of the FAA's review, a small aircraft crashed during a missed approach to a regional airport in Georgia. The NTSB found that, although the airport tower received four MSAW warnings concerning the aircraft, the controller did not notice the warnings because he was attending to other duties. The MSAW system installed at the tower was configured for visual alerting only, and the controller was not monitoring the screen on which the warnings were being displayed. The NTSB concluded that the controller would have noticed the warnings if they had been issued aurally as well as visually, and asked the FAA to "require the installation of aural [MSAW] equipment [in facilities] that would otherwise receive only a visual alert" (NTSB, 2001).

On October 2, 1996, a small aircraft crashed while on approach to a regional airport in Maryland. The controllers at the regional ATC facility stated that they did not see or hear any MSAW alerts concerning the aircraft. Upon investigating the facility, the NTSB discovered that the MSAW aural alarm speaker "was covered with heavy paper held in place with what appeared to be masking tape," apparently because the system had frequently been generating nuisance warnings (NTSB, 2001). The NTSB recommended that the FAA immediately inspect its air traffic facilities for muted MSAW speakers, train controllers on the nature of MSAW and how to respond to MSAW alerts, modify the MSAW system to enhance the conspicuity of jeopardized aircraft, and, in its upcoming STARS ATM system, include a MSAW aural alert speaker at each radar display. The FAA has complied with some of these recommendations; others are still ongoing.

On August 6, 1997, Korean Air flight 801, a Boeing 747 carrying 254 people, crashed on approach to Guam International Airport, killing 228 and injuring 26. During its investigation, the NTSB learned that the FAA had intentionally inhibited the Guam MSAW system in order to eliminate

nuisance alarms. If the system had been operational, it would have issued an alert concerning Korean Air 801 64 seconds before impact. In response to this accident, the FAA recertified all of its MSAW installations and revised its standards and guidelines for configuring MSAW systems. It also developed a comprehensive program to validate MSAW site configurations as part of its commissioning process for new air traffic facilities.

On January 13, 1998, a Learjet crashed during approach at Houston Intercontinental Airport in Texas. The NTSB's investigation revealed yet another instance in which the MSAW system was configured improperly in spite of the guidelines the FAA produced in response to the Guam accident.

With these various incidents in mind, in the next section we hypothesize a series of revised safety cases that might have been created for this system had the FAA documented its safety argument rigorously. We also argue that, had they followed the enhanced safety-case lifecycle, many of these accidents could have been avoided.

### 4.3 MSAW System Safety Argument

Even though the MSAW system was not certified using evidence-based assurance, it was developed, augmented, and revised through the analysis of failures, or more specifically, aircraft accidents. Thus, it exemplifies how the enhanced safety-case lifecycle could be put into practice. The MSAW failures involved in the incidents described above can be classified into two major categories: (1) those in which MSAW did not generate an alert because it was configured improperly; and (2) those in which MSAW generated an alert but the alert failed to notify the controllers. This section considers how the latter set of incidents would have affected the MSAW safety argument had the FAA and the NTSB employed the lifecycle we discussed in Section 3. The accident in which the MSAW speakers were muted, although belonging to this category, is outside our scope of consideration.

Before MSAW, the safety argument that ATC would detect low-flying aircraft relied on the assumption that controllers would be able to manually spot such aircraft on their radar displays. This argument is illustrated in GSN in Fig. 3.

Upon investigating the 1972 accident, the NTSB would have learned that the justification J1 was invalid because merely displaying the altitude data was not sufficient to alert the controller of low-flying aircraft and because controllers were not always aware of the terrain elevations for each region of their radar displays. Therefore, the *lesson* from this accident, stated according to the guidelines from Section 3.2, would be, "The justification that the controller is aware of the regional terrain and will manually identify

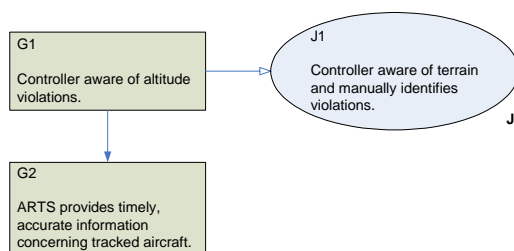


Figure 3. Assumed Original Implicit Argument for Controller Awareness

altitude violations (J1) is invalid.” It is clear that without this justification, the original safety argument is invalidated because satisfying subgoal G2 no longer satisfies goal G1. The recommendation from this accident, stated as a revision of the invalidated justification, would be, “Issuing an automated alert whenever an altitude violation occurs would be sufficient to notify the controller of the violation.” In order to use this new justification to rebuild the safety argument, the developer (in this case, the FAA) would now have to show that the system provides such an alert. Showing this property would have required the FAA to change the system by implementing MSAW, resulting in the new safety argument depicted in Fig. 4.

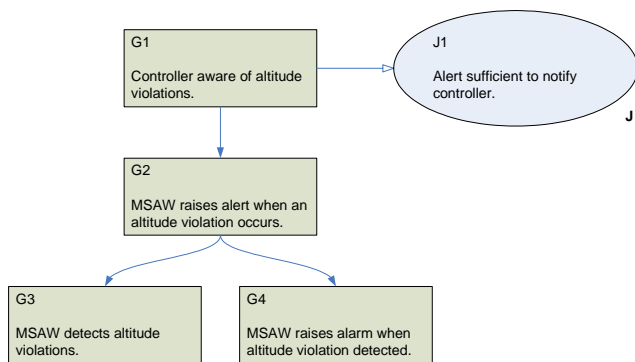


Figure 4. Assumed Original MSAW Controller Awareness Argument

The revised safety argument addresses the lessons and recommendations from the 1972 accident, but it is still flawed. Justification J1, which was the recommendation from the 1972 accident investigation, is vague because it does not precisely define what an alert should be. Consequently, the justification can be used to show that G2 implies G1 irrespective of the type of alert used. The NTSB discovered this ambiguity in its investigation of the 1995 Georgia accident when it learned that a visual alert alone would not always attract the controller’s attention. The lesson from that accident would

once again be that J1 is invalid because the meaning of “alert” is ambiguous. The justification should read, “Visual *and aural* alerts are sufficient to notify controllers of altitude violations.” Alternatively, contextual information could be included to define an alert as comprising both visual and aural components.

As a result of the lessons from the 1995 investigation, the MSAW safety argument was once again invalidated. In order to rebuild the argument this time, the FAA would now have to show that the system provided visual and aural alerts requiring the installation of speakers at each ATC facility. The final hypothetical argument is shown in Fig. 5, which includes revisions necessitated by the lessons learned from the configuration-related MSAW accidents such as the 1997 Guam accident. Subgoals G5, G6, and G7 are placeholders and would need to be developed further in an actual argument.

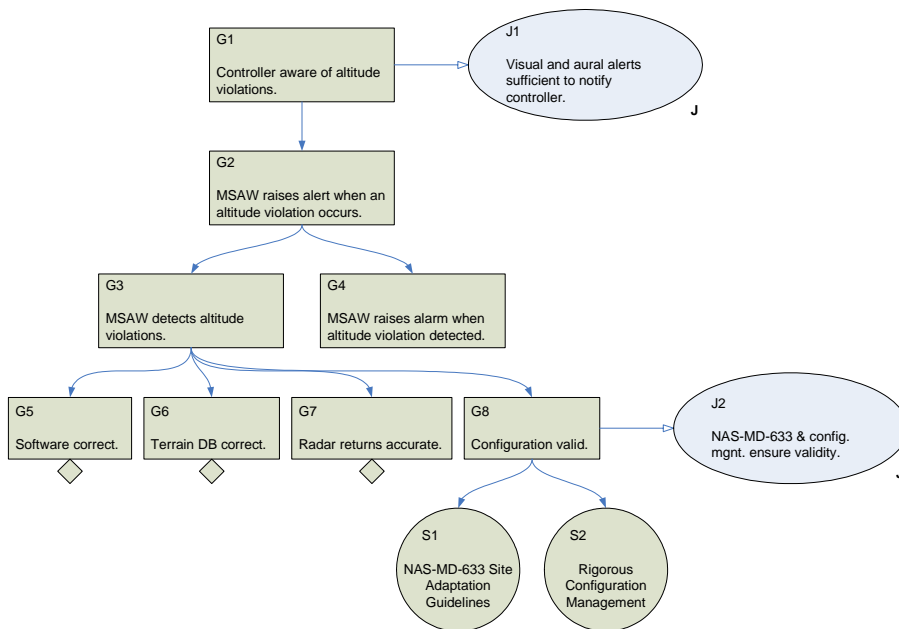


Figure 5. Revised MSAW Controller Awareness Argument

## 4.4 Observations

Seven accidents occurred between the time the need for a minimum safe altitude warning system was realized and the time at which its safety argument reached its current state. Of these accidents, only three provided new information that could not have been inferred from prior failures. The

1972 accident established the need for MSAW, the 1989 accident showed that configuration management played a crucial role in the system's ability to detect altitude violations, and the 1995 accident revealed that aural as well as visual alerts were needed to notify controllers of violations. We hypothesize, but we cannot claim, that had the NTSB and FAA used the enhanced safety-case lifecycle we propose, at most these three accidents would have occurred before the safety argument shown in Fig. 5 would have been reached.

Our justification for this statement is that the first-class status of each argument element in the safety case would have required investigators to examine each element associated with the defective part of the pre-failure safety case. By having to document and justify the revised assumption in the post-failure safety case concerning MSAW alerting (which turned out to be flawed), investigators would have been more likely to detect the ambiguity in the meaning of "alert" that contributed to the 1995 accident. The probability of detection might have been raised further if the difficulty of obtaining proper shared meaning of terms like "alert" had been addressed (Hanks, 2003).

One notable difference between the NTSB's safety recommendations and our lessons and recommendations lies in how change is effected. In the former, the call for change is made in the safety recommendation, which usually suggests specific design or procedural changes. These changes might be infeasible for various reasons, and so they may be rejected by the system developer (as actually happened in this case, where the developer was the FAA). In the lifecycle we propose, the call for change is made in the lessons, which are findings of fact evidenced by the failure itself. They are separated from the recommendations, which suggest possible means, but not the only means, of revising the safety argument in light of the new lessons. The validity of the lessons is independent of that of the recommendations, and so the call for change remains intact even if a developer disagrees with the recommendations that accompany it.

## 5. CONCLUSION

Developers of safety-critical systems and those who investigate the failures of such systems share a common goal of ensuring that the system does, in operation, meet the goals set out in its safety case. This assurance process is most efficient and effective when the relationship between a system's operation and the case for its safety is clearly stated and exploited. We have developed and illustrated a method for using the safety case to assist in failure analysis. This method greatly improves our assurance of

system safety because it augments traditional safety-case development and maintenance activities to include explicit feedback through analysis of the system's failures in operation.

## ACKNOWLEDGEMENTS

This work was funded in part by NASA Langley Research Center under grants numbered NAG-1-2290 and NAG-1-02103.

## REFERENCES

- Bishop, P.G., and R. E. Bloomfield. "A Methodology for Safety Case Development." Safety-critical Systems Symposium, Birmingham, UK, Feb 1998.
- Hamilton, V. and Rees, C. "Safety Integrity Levels: An Industrial Viewpoint." *Towards System Safety: Proc. Seventh Safety Critical Systems Symposium*, Huntington, U.K. T. Anderson and F. Redmill (Eds.). Springer-Verlag. 1999.
- Hanks, Kimberly S., and John C. Knight. "Improving Communication of Critical Domain Knowledge in High-Consequence Software Development: An Empirical Study." 21st International System Safety Conference, Ottawa, Canada, August 2003.
- Johnson, C.W. *Failure in Safety-Critical Systems: A Handbook of Accident and Incident Reporting*. Glasgow: University of Glasgow Press. 2003.
- Kelly, T.P. "Arguing Safety—A Systematic Approach to Managing Safety Cases," Ph.D. diss. University of York. 1998.
- McDermid, J.A. "Software Safety: Where's the Evidence?" *Proc. 6<sup>th</sup> Australian Workshop on Industrial Experience with Safety Critical Systems and Software (SCS '01)*, Brisbane, Australia. P. Lindsay (Ed.). Conference in Research and Practice in Information Technology, Vol. 3. 2001.
- McDermid, J.A. and Kelly, Tim. "A Systematic Approach to Safety Case Maintenance." *Proc. International Conference on Computer Safety, Reliability, and Security (SAFECOMP '99)*, Toulouse, France. 1999.
- National Transportation Safety Board. *Controlled Flight Into Terrain, Korean Air Flight 801, Boeing 747-300, HL7468, Nimitz Hill, Guam, August 6, 1997*. Aircraft Accident Report NTSB/AAR-00/01. Washington, DC, 2000.
- RTCA. "Software Considerations in Airborne Systems and Equipment Certification," document RTCA/DO-178B. Washington, DC: RTCA, December 1992.
- Weaver, R.A. "The Safety of Software: Constructing and Assuring Arguments," Ph.D. diss. University of York. 2003.