Designation: F3269 – 21

Standard Practice for Methods to Safely Bound Behavior of Aircraft Systems Containing Complex Functions Using Run-Time Assurance[1]

This standard is issued under the fixed designation F3269; the number immediately following the designation indicates the year of original adoption or, in the case of revision, the year of last revision. A number in parentheses indicates the year of last reapproval. A superscript epsilon (ε) indicates an editorial change since the last revision or reapproval.

INTRODUCTION

This practice defines an architecture using Run-Time Assurance (RTA) in conjunction with unassured functions or commercial off-the-shelf (COTS) functions that have not been developed to traditional aerospace standards and processes. This section provides the scope, applicability, and intended use for the understanding of this practice.

The practice is organized as follows: *(1)* An introduction, background, and scope to provide context for applying the capabilities defined in this practice to unmanned aircraft system (UAS) certification, or operational approval, or both. *(2)* Definitions of key terms and abbreviations. *(3)* Description of a Run-Time Assurance (RTA) architecture. *(4)* Appendixes that contain Examples of RTA in systems and supplemental information. *(a)* Ground Collision Avoidance System (GCAS) as an Example RTA. *(b)* Machine Learning AI Autopilot (MLAA). *(c)* Run-Time Assurance for a Neural Network-Based Adaptive Flight Control of an Unmanned Aircraft. *(d)* Run-Time Assurance for Risk-Based Operation. *(e)* Example Implementation of Timing and Latency Requirement. *(5)* A list of documents referenced herein.

BACKGROUND

There is significant interest from industry and civil aviation authorities (CAA) to have a standard practice to enable new and novel technologies used in UAS operations containing unassured or COTS functions/systems, or both, to be used on certified aircraft and aviation systems. From this point forward, "functions/systems" will be referenced as "functions." Developing a certification path for these technologies may also introduce greater safety to aviation.

In this practice, the term *Complex Function* (CF) may be any function, algorithm, component, or system that has not been subject to accepted CAA or aerospace design assurance practices, or both (DO-178C, DO-254, ARP4754A, etc.). Motivations to use such an unassured function arise from the need or desire to use commercial, off-the-shelf systems or parts that have algorithmic complexity, probabilistic algorithms, fuzzy logic, environmental uncertainties, or no pedigree. The complexity may also come from factors associated with new and novel technologies such as sensor measurement precision, nondeterministic algorithms, data-driven algorithms, or artificial intelligence (for example, machine learning, genetic algorithms). A complex function may be any combination of software or hardware.

Traditional approaches to digital avionics design begin with the assumption that each software and hardware component on an aircraft contribute independently to the safe operation of the platform. At the core of this process is an assessment of the risks associated with the functional failure of each system, assembly, or component to ensure that the aircraft meets the required safety objectives. This is known as design-time assurance.

This practice describes a run-time assurance method, which may be used as an alternative means to or in combination with design-time assurance. RTA mitigates the risk of complex function misbehavior by managing the system's use of the Complex Function output. The RTA includes a safety monitor, which monitors the complex function or the behavior the complex function has on the system, or both, at run-time. In the event the safety monitor determines that the complex function is not operating correctly, or is driving the system to an unsafe state, it disengages the complex function and initiates a recovery function.

This practice provides an RTA architecture and best practices that provide guidance to an applicant for ensuring that the behavior of an unmanned aircraft system (UAS) containing complex functions maintains the acceptable level of safety.

At the time of this practice's development, there is no accepted formal guidance material for certifying commercial UAS containing complex functions. Emerging CAA certification guidance, processes, and concepts have been considered in the development of this practice.

[1] This practice is under the jurisdiction of ASTM Committee F38 on Unmanned Aircraft Systems and is the direct responsibility of Subcommittee F38.01 on Airworthiness.
Current edition approved July 15, 2021. Published November 2021. Originally approved in 2017. Last previous edition approved in 2017 as F3269–17. DOI: 10.1520/F3269-21.

1. Scope

1.1 The scope of this practice includes the following:

1.1.1 A set of components that comprise an RTA system.

1.1.2 Requirements and best practices to determine safe boundaries and RTA system coverage.

1.1.3 Requirements and best practices for an RTA system and RTA components, as applicable.

1.1.4 Appendixes with examples that demonstrate key RTA system concepts.

1.2 RTA components are required to meet the design assurance level dictated by a safety assessment process. Guidance for the safety assessment process may be found in references appropriate for the intended operations (ARP4754A, ARP4761, Practice F3178, etc.).

1.3 This practice was developed with UAS in mind. It may be applicable for aspects of manned aircraft certification/approval, as well as aviation ground systems. The scope of this practice is also envisioned to allow a variety of aircraft implementations where a human may perform the role of either the Complex Function or a Recovery Function.

1.4 The scope of this practice does not cover aspects of hardware/software integration. These should be considered separately during the development process.

NOTE 1—This practice does not suggest a one-size-fits-all strategy knowing that not all use cases may fit well into this architecture. There may exist additional components required to satisfy specific applications to the practice.

1.5 The values stated in inch-pound units are to be regarded as standard. No other units of measurement are included in this standard.

1.6 *Table of Contents:*

1.7 *This standard does not purport to address all of the safety concerns, if any, associated with its use. It is the responsibility of the user of this standard to establish appropriate safety, health, and environmental practices and determine the applicability of regulatory limitations prior to use.*

1.8 *This international standard was developed in accordance with internationally recognized principles on standardization established in the Decision on Principles for the Development of International Standards, Guides and Recommendations issued by the World Trade Organization Technical Barriers to Trade (TBT) Committee.*

2. Referenced Documents

2.1 *ASTM Standards:*[2]

F3060 Terminology for Aircraft

F3178 Practice for Operational Risk Assessment of Small Unmanned Aircraft Systems (sUAS)

F3341/F3341M Terminology for Unmanned Aircraft Systems

ASTM AC377 TR2-EB Developmental Pillars of Increased Autonomy for Aircraft Systems

2.2 *FAA Advisory Circular:*[3]

AC 23.1309-1E System Safety Analysis and Assessment for Part 23 Airplanes

2.3 *RTCA Standards:*[4]

RTCA DO-178C Software Considerations in Airborne Systems and Equipment Certification

RTCA DO-254 Design Assurance Guidance for Airborne Electric Hardware

2.4 *SAE Standards:*[5]

SAE ARP4754A Guidelines for Development of Civil Aircraft and Systems

SAE ARP4761 Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment

3. Terminology

3.1 This section defines key terms and abbreviations for this practice.

3.2 Note on terminology: *shall versus should versus may*—use of the word "shall" implies that a procedure or statement is mandatory and must be followed to comply with this practice, "should" implies recommended, and "may" implies optional at the discretion of the supplier, manufacturer, or operator. Since "shall" statements are requirements, they include sufficient detail needed to define compliance (for example, threshold values, test methods, oversight, and references to acceptable industry standards). "Should" statements represent best practices to guide in the development of RTA Systems. "May" statements are provided to clarify acceptability of a specific item or practice and offer options for satisfying requirements.

3.3 *Unique and Common Terminology*—Terminology used in multiple standards is defined in F3341/F3341M, UAS Terminology Standard, and F3060, Aircraft Terminology Standard.

3.4 *Definitions of Terms Specific to This Standard:*

3.4.1 *Assured, adj*—Attribute of an entity for which sufficient evidence exists to demonstrate that an acceptable level of rigor has been met.

3.4.2 *Complex Function, n*—the unassured function for which run-time-assurance is being used. For examples, see *Background*.

3.4.3 *Designer, n*—the person or organization that is responsible for the design, development, and/or integration of the RTA System.

3.4.4 *Dynamic Consistency, n*—independently measured variables are checked for consistency using known models of behavior. For further detail, reference ASTM AC377 TR2-EB, Section 5, Dynamic Consistency.

3.4.5 *Input Manager, n*—an assured RTA function that accepts assured and unassured data and conditions, validates and performs consistency checking, and outputs assured data to RTA Components.

3.4.6 *Larger System, n*—the system within which the RTA System exists. It provides external RTA data and inputs and consumes the RTA Output. *Example of Larger Systems are avionics system/subsystem, air vehicle, or UAS, which may contain multiple RTA Systems.*

3.4.7 *Larger System Specification, n*—The collection of all requirements used to specify the design of the Larger System. A subset of the Larger System Specification contains requirements derived from this architecture standard specifying the design and implementation of the RTA System.

3.4.8 *Monitor Coverage, n*—union of the coverage provided by each Monitor Subfunction within the RTA system.

3.4.9 *Predefined Bounds, n*—acceptable limits to maintain the Larger System in a Safe State. Any violation of this bound is a failure of the RTA System. *Predefined Bounds may be static or dynamic and are determined during design.*

3.4.10 *Recovery Function, n*—an assured RTA function that generates outputs intended to keep the Larger System in a Safe State. *Recovery Function may provide "fail safe" or "fail functional" capabilities in order to allow for graceful degradation of functionality.*

3.4.11 *Recovery Function Coverage, n*—union of the coverage provided by each Recovery Function within the RTA System.

3.4.12 *RTA Components, n*—the set of assured functions defined by RTA architecture; includes Input Manager, Safety Monitor, RTA Switch, Recovery Function(s).

3.4.13 *RTA Output, n*—the output of the RTA Switch.

3.4.14 *RTA Switch, n*—an assured RTA function that accepts the source selection for the RTA Output from the Safety Monitor and provides that one output to the Larger System.

3.4.15 *RTA System, n*—the system containing RTA Components and the Complex Function.

3.4.16 *RTA System Coverage, n*—the RTA System's operational domain where both Monitor Coverage and Recovery Coverage exists.

3.4.17 *Run-Time Assurance, n*—a method that uses RTA systems to ensure that a Larger System's behavior remains in a Safe State.

[2] For referenced ASTM standards, visit the ASTM website, www.astm.org, or contact ASTM Customer Service at service@astm.org. For *Annual Book of ASTM Standards* volume information, refer to the standard's Document Summary page on the ASTM website.

[3] Available from Federal Aviation Administration (FAA), 800 Independence Ave., SW, Washington, DC 20591, http://www.faa.gov.

[4] Available from Radio Technical Commission for Aeronautics (RTCA), 1150 18th NW, Suite 910, Washington, DC 20036, https://www.rtca.org.

[5] Available from SAE International (SAE), 400 Commonwealth Dr., Warrendale, PA 15096, https://www.sae.org.

3.4.18 *Run-Time Assurance Architecture, n*—a system of assured components that implements monitoring, prediction, and fail-safe recovery mechanisms that bounds the behavior of a system containing a Complex Function. RTA Components are: Input Manager, Safety Monitor, RTA Switch, and Recovery Function(s).

3.4.19 *Safe State, n*—a condition where the Larger System is within acceptable limits.

3.4.20 *Safety Assessment Process, n*—the set of activities applied during the design of the Larger System to generate safety objectives and determine the necessary level of assurance for the RTA Components.

3.4.21 *Safety Monitor, n*—an assured RTA function that continuously evaluates Larger System and/or Complex Function behaviors, with the intent of discovering misbehavior of the Complex Function. When necessary, the monitor selects and commands the RTA Switch to a Recovery Function or back to the Complex Function. The Safety Monitor is composed of one or more Monitor Subfunctions.

3.4.22 *Safety Monitor Trigger Threshold (SMTT), n*—limits derived from the Predefined Bounds that are used by the Safety Monitor to determine the source of the RTA Output. *The SMTT may be static or dynamic and is determined during design.*

3.4.23 *Unassured, adj*—Attribute of an entity that is not assured and, hence, may not be directly used and trusted by RTA components.

3.5 *Abbreviations:*

3.5.1 *CAA*—Civil Aviation Authority

3.5.2 *CF*—Complex Function

3.5.3 *IM*—Input Manager

3.5.4 *RF*—Recovery Function

3.5.5 *RS*—RTA Switch

3.5.6 *RTA*—Run-time assurance

3.5.7 *SM*—Safety Monitor

3.5.8 *SMTT*—Safety Monitor Trigger Threshold

3.5.9 *UAS*—Unmanned Aircraft System

4. Significance and Use

4.1 This practice provides an architectural framework for developing an RTA system, which provides run-time assurance as an alternative to design-time assurance to fulfill safety requirements for an unassured or complex function. The standard provides best practices and guidelines to assist in the RTA system's development. Further, it describes the architectural components and requirements for designing the RTA system. Compliance to this practice is achieved by deriving RTA System requirements from the standard and capturing them in the Larger System Specification. The system design requirements can then be validated and verified using acceptable engineering practices. It is anticipated that this practice will provide a means to accept complex automation/autonomy aircraft functions that have been difficult to certify using traditional methods.

4.2 The following three-step process is used to derive verifiable design requirements using this architecture standard:

4.2.1 Create RTA System requirements using the guidance provided by this architecture standard.

4.2.2 Capture RTA System requirements in the Larger System Specification.

4.2.3 Perform verification and validation on the RTA System requirements in the Larger System Specification.

4.3 The RTA architecture can be applied to all sizes, levels, and classes of UAS. Using run-time assurance can provide systems with the following benefits:

4.3.1 The ability to mitigate hazards related to nondeterministic or unexpected behavior from unassured functions that employ advanced software methods or algorithmic complexity that cannot be certified using traditional certification practices.

4.3.2 The ability to use functions for which it may not be possible to obtain artifacts of conventional DO-178 or DO-254 assurance processes.

4.3.3 The ability to use COTS hardware or software, or both, for the unassured function.

4.3.3.1 For example, automotive components, thereby leveraging mature software with extensive service history that was developed for other safety-critical industries, but cannot be shown to comply with aviation development assurance practices.

4.3.3.2 For example, industry components where source code or other associated engineering artifacts are unavailable.

4.3.4 A reduction in cost and schedule burdens by allowing rapid design iterations of the unassured or complex function during and after initial certification. This update of the standard allows unassured or complex function upgrades after initial certification to minimize subsequent modifications to the certification or approval.

5. RTA Functional Architecture

5.1 This section defines key attributes of the overall RTA architecture and its components that meet the intent described in this practice. Minimum requirements are defined for various intended uses of this reference architecture (see Fig. 1). It is expected that the reference architecture will be tailored by the users to their specific application.

5.2 Subsections 5.3 through 5.9 are written to solely provide the functional characteristics of an RTA System. The RTA components with their attributes and their relationships are described in 5.3 through 5.9. Implementations may distribute RTA functionality across hardware and software modules, as desired.

5.3 This practice is written with a single RTA System in mind, that is, where a Larger System's behavior is bound using a single RTA implementation. However, complex systems containing multiple independent RTA systems are envisioned. This practice is applicable to multiple, composable RTA systems as long as their respective RTA Outputs are independent (that is, RTA Systems do not contend to output the same data).

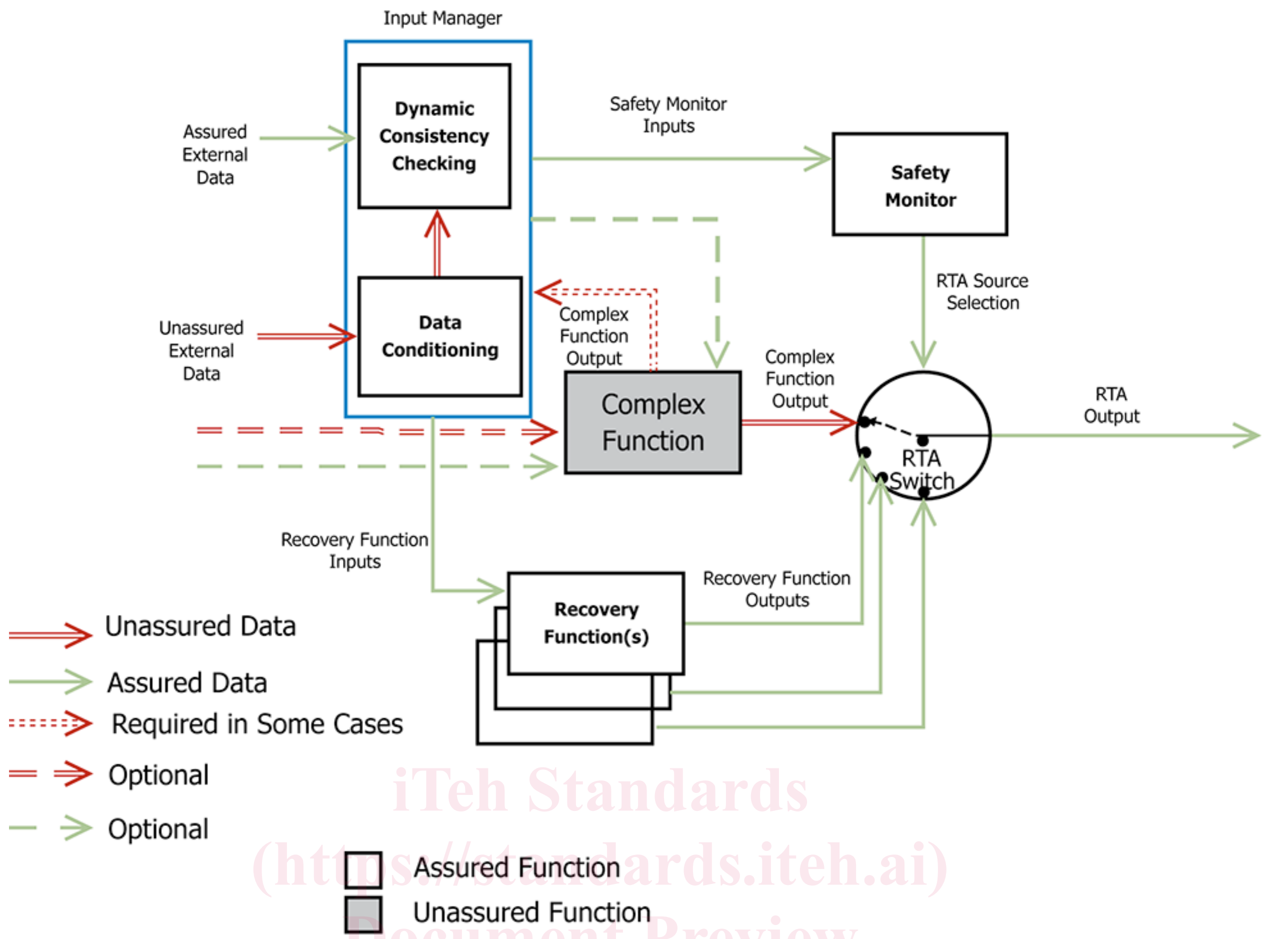

FIG. 1 RTA Architecture

5.4 *Overall Architecture:*

5.4.1 *Components and Interfaces:*

5.4.1.1 The RTA System architecture (Fig. 1) consists of the following Components and Interfaces:

(1) RTA Components
- *(a)* Input Manager
  – Data Conditioning
  – Dynamic Consistency Checking
- *(b)* Safety Monitor
- *(c)* RTA Switch
- *(d)* Recovery Function(s)

(2) RTA Interfaces
- *(a)* Assured External Data
- *(b)* Unassured External Data
- *(c)* Safety Monitor Inputs
- *(d)* Recovery Function Inputs
- *(e)* Recovery Function Outputs
- *(f)* Complex Function Outputs
- *(g)* RTA Source Selection
- *(h)* RTA Output

(3) Complex Function

5.4.1.2 This section provides guidance to the developer for implementing an RTA System. The architecture ensures that "RTA Output" in Fig. 1 is always bounded with the intent that the Larger System containing the Complex Function remains in a Safe State. First, the Unassured External Data is conditioned, then verified for dynamic consistency. Assured External Data does not require conditioning. Outputs from the Input Manager can be safely used within the RTA System. The Safety Monitor ensures that the RTA Output or Larger System behavior, or both, is within Predefined Bounds. When the Safety Monitor determines that the RTA Output or Larger System behavior, or both, exceeds the Safety Monitor Trigger Threshold, the Safety Monitor selects an appropriate Recovery Function and passes the selection to the RTA Switch. The RTA Switch then ensures that the RTA Output is sourced from the selected Recovery Function. The Safety Monitor may return control to the Complex Function when it is determined that the Complex Function's behavior has become acceptable.

NOTE 2—When monitoring the Complex Function behavior directly (as opposed to monitoring the behavior of the Larger System), the Complex Function Output is routed to the Safety Monitor through the Input Manager. This allows data conditioning and dynamic consistency checking of the Complex Function Output while maintaining its behavioral attributes.

5.4.2 *RTA System Coverage:*

5.4.2.1 This section describes one process for defining *RTA System Coverage*. Determining coverage for an RTA System is highly dependent on the use-case and pertinent variables. Fig. 2 denotes an abstract 2-D version of coverage for illustrative purposes.

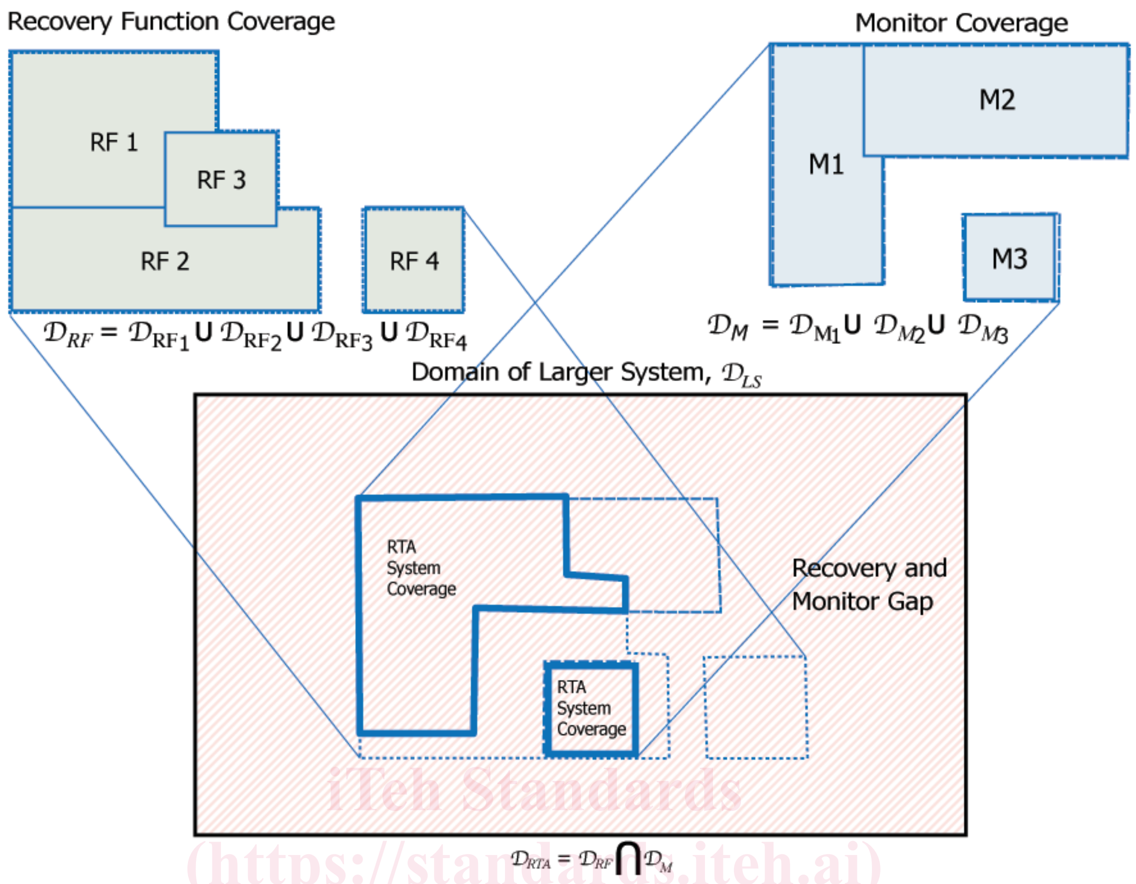
Recovery Function Coverage
RF 1
RF 3
RF 2
RF 4
$\mathcal{D}_{RF} = \mathcal{D}_{RF1} \cup \mathcal{D}_{RF2} \cup \mathcal{D}_{RF3} \cup \mathcal{D}_{RF4}$
Monitor Coverage
M2
M1
M3
$\mathcal{D}_M = \mathcal{D}_{M1} \cup \mathcal{D}_{M2} \cup \mathcal{D}_{M3}$
Domain of Larger System, $\mathcal{D}_{LS}$
RTA System Coverage
RTA System Coverage
Recovery and Monitor Gap
$\mathcal{D}_{RTA} = \mathcal{D}_{RF} \cap \mathcal{D}_M$

FIG. 2 RTA System Coverage

5.4.2.2 The RTA System Coverage may be determined as follows:

(1) Denote D_{LS} as the set of all reachable, nominal, or otherwise, operational points where the Larger System is expected to operate.

(2) For every available Recovery Function, RF_i, compute the set of all points that can be design-time assured to maintain Larger System safety or recover functionality, or both, provided by the Complex Function at those points. Denote this computed set as D_{RF_i}.

(3) For every available Monitor Function, M_j, compute the set of all points that can be design-time assured to detect Complex Function or Larger System misbehavior. Denote this computed set as D_{M_j}.

5.4.2.3 Then the RTA System Coverage may be defined as:

$$D_{RTA} = \left(\bigcup_{i=1}^{n_{RF}} D_{RF_i} \right) \cap \left(\bigcup_{j=1}^{n_M} D_{M_j} \right) = D_{RF} \cap D_M \quad (1)$$

where:

n_{RF}, n_M	=	the number of available recovery functions and monitor functions, respectively; and
∪ *and* ∩	=	the union and intersection operators, respectively.

Note that Eq 1 shows that RTA System Coverage can be composed of disconnected sets.

5.4.2.4 Larger System behavior is only bounded by the RTA System when operating within D_{RTA}. Therefore, operation of the Larger System outside the RTA System Coverage is outside the scope of this practice.

5.4.3 *RTA Scenarios:*

5.4.3.1 Four possible scenarios are identified in Fig. 3. All scenarios start with *(1)* the Complex Function is the source of the RTA Output, *(2)* the RTA Output is within the Safety Monitor Trigger Threshold, which is defined as the RTA System being within the *Nominal Region*.

5.4.3.2 When the RTA Output exceeds the Safety Monitor Trigger Threshold and is within the Predefined Bounds, the RTA System is defined as being in the *Recovery Region*. The RTA Output is sourced from one of the Recovery Functions. If the RTA Output exits the Predefined Bounds, the RTA System has failed.

5.4.3.3 *Scenario a)*—RTA Output remains within the Safety Monitor Trigger Threshold, and the RTA System remains within the Nominal Region. The Complex Function remains the source of the RTA Output.

5.4.3.4 *Scenario b)*—RTA Output exceeds the Safety Monitor Trigger Threshold, thus, the RTA System exits the Nominal Region. A Recovery Function becomes the source of the RTA Output. When the Complex Function Output returns to within the Safety Monitor Trigger Threshold, the RTA System re-enters the Nominal Region, and the Complex Function Output again becomes the source of RTA Output.

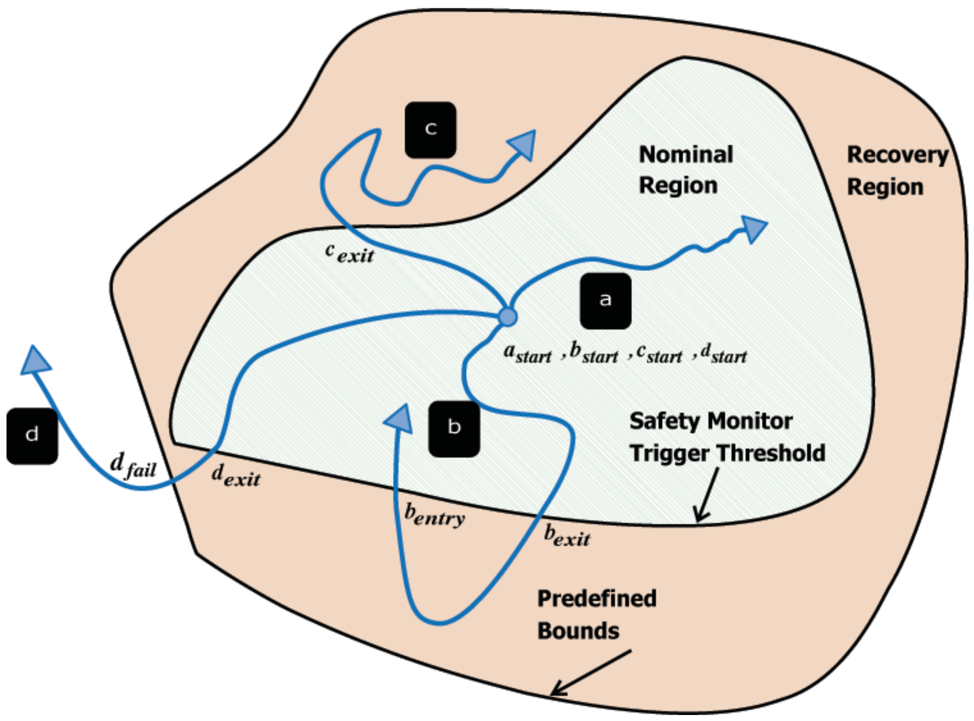

FIG. 3 RTA System Operational Scenarios

5.4.3.5 *Scenario c)*—RTA Output exceeds the Safety Monitor Trigger Threshold, thus, the RTA System exits the Nominal Region. A Recovery Function becomes the source of the RTA Output. The Recovery Function is unable to return the RTA System to the Nominal Region, or the Complex Function Output continues to exceed the Safety Monitor Trigger Threshold, or both. The RTA Output remains sourced from the Recovery Function, and the RTA System operates safely within the Recovery Region.

5.4.3.6 *Scenario d)*—RTA Output exceeds the Safety Monitor Trigger Threshold, thus, the RTA System exits the Nominal Region. A Recovery Function becomes the source of the RTA Output. The Recovery Function is unable to return the RTA Output to within the Safety Monitor Trigger Threshold or even keep it within the Predefined Bounds. The RTA System exits the Recovery Region, thus, the RTA System has failed.

5.4.3.7 For all scenarios, the Safety Monitor Trigger Threshold and Predefined Bounds may be either static or dynamic. Dynamic bounds and thresholds require continual recomputation to determine the actual boundary of the Recovery Region. Dynamically changing bounds and thresholds may be imprecise and may benefit from or even require implementation of a safety margin to ensure the RTA System does not accidentally exit the Recovery Region.

5.4.3.8 *Event Sequencing and Timing:*

(1) For each of the scenarios in Fig. 3, the following sequence of events are possible:

a_{start}
$b_{start} \rightarrow b_{exit} \rightarrow b_{entry}$
$c_{start} \rightarrow c_{exit}$
$d_{start} \rightarrow d_{exit} \rightarrow d_{fail}$

where the times at which the *start*, *exit*, *entry*, and *fail* events occur for each of the scenarios are ordered according to $t_{start} \leq t_{exit} \leq t_{entry} \leq t_{fail}$. All events can occur at the same time step, but in the order afforded above.

(2) Even in *Scenario Case a)* where the RTA System is within the Nominal Region, an estimate of when and whether the a_{exit} and a_{fail} may occur should be used to compute buffers, margins, and other quantities that may be required to make decisions to maintain the RTA System within the Nominal Region.

5.4.4 *Best Practices:*

5.4.4.1 *Safety Assessment Process*—The designer should apply a Safety Assessment Process to determine the functional hazard associated with a failure of the Complex Function. The hazardous conditions associated with a failure of the Complex Function are used to determine the necessary levels of design assurance imposed on the RTA components.

5.4.4.2 *Robust Design*—The designer should engage in appropriate design, test, and validation activities to enable the complex function and RTA components to perform as intended, including establishing required resources, bandwidth, etc.

5.4.4.3 *Poor Complex Function Design*—The designer should not use RTA as a substitute for poor complex function design or implementation, or both.

5.4.4.4 *False Recovery Activation*—The designer should define the Safety Monitor Trigger Threshold such that the RTA System has an acceptable false alarm rate. The false alarm rate should be defined in the Larger System Specification.

5.4.4.5 *Chattering*—The designer should prevent the occurrence of frequent switching between Complex Function and Recovery Functions.

5.4.4.6 *Partitioning and Modularity*—The designer should leverage the concepts of partitioning and modularity when developing the RTA System.

5.4.4.7 *Safety Margins*—The designer should ensure RTA activates in a timely manner. Safety Margins should be selected to ensure boundaries are not violated. For example, margins may be considered in the definitions of *Predefined Bounds* and *Safety Monitor Trigger Threshold*.

5.4.4.8 *Recovery Function Monitoring*—The designer should consider availability of Recovery Functions in RTA System design.

5.4.5 *Requirements:*

5.4.5.1 *F3269-RTAS-001*—The designer shall develop RTA Components to the necessary level of assurance as determined in the Larger System Specification.

5.4.5.2 *F3269-RTAS-002*—The designer shall develop all RTA components and RTA interfaces listed in 5.4.1.

5.4.5.3 *F3269-RTAS-003*—The designer shall quantify and address the impact of timing and latency in the RTA System and the Larger System on each RTA Component. *This includes all timing considerations such as latency, time constants, and event sequencing from sensor input to detecting misbehavior through completion of the recovery process.*

5.4.5.4 *F3269-RTAS-004*—The designer shall ensure that adverse transients do not occur while switching between different sources of RTA Output.

5.4.5.5 *F3269-RTAS-005*—The designer shall determine RTA System Coverage in accordance with the process defined in 5.4.2.

5.4.5.6 *F3269-RTAS-006*—The designer shall ensure that the RTA System is only used within the computed RTA System Coverage.

5.4.5.7 *F3269-RTAS-007*—The designer shall define bounds to ensure that the Larger System remains in a Safe State.

5.4.5.8 *F3269-RTAS-008*—The designer shall define a threshold beyond which the Safety Monitor activates a Recovery Function that ensures the RTA Output or Larger System behavior, or both, is maintained within Predefined Bounds. *The overall recovery process includes all activities and their timing, from exit of the region enclosed by the SMTT through re-entry; this may include determination that the trigger threshold has been exceeded, time to select a recovery function, time to initialize the Recovery Function, time to switch to the Recovery Function, time to perform recovery, and time to re-enter the region enclosed by the SMTT, when possible.*

5.4.5.9 *F3269-RTAS-009*—The designer shall ensure RTA Component failures are detected and handled.

5.5 *RTA Interfaces:*

5.5.1 *Assured External Data*—Interface from Larger System to the RTA System that contains only assured data.

5.5.2 *Unassured External Data*—Interface from the Larger System to the RTA System that may contain unassured data.

5.5.3 *Safety Monitor Inputs*—Interface from the Input Manager to the Safety Monitor that includes the minimum set of assured data needed to monitor Larger System or Complex Function behavior, or both.

5.5.4 *Recovery Function Inputs*—Interface from the Input Manager to the Recovery Function that includes a minimum set of assured data needed to perform its intended function.

5.5.5 *Complex Function Output*—Interface from the Complex Function to the RTA Switch and the Input Manager when monitoring the behavior of the Complex Function. *It is the desired source for the RTA Output.*

5.5.6 *Recovery Function Outputs*—Interface from the Recovery Function to the RTA Switch that contains only assured data. It is an alternate source for the RTA Output.

5.5.7 *RTA Source Selection*—Interface from the Safety Monitor to the RTA Switch that contains only assured data, which specifies the source of the RTA Output.

5.5.8 *RTA Output*—Interface from the RTA Switch to the Larger System. The RTA Output has been bounded by the RTA System.

5.6 *Input Manager:*

5.6.1 *Description:*

5.6.1.1 The Input Manager's role is to ensure that downstream RTA components receive assured data. This is accomplished using two distinct capabilities—data conditioning and dynamic consistency checking. Embedded systems data can be invalid for a variety of reasons. It is important to perform data validity checking (or data conditioning and dynamic consistency checking). In the following paragraphs, we discuss the means to show that data is assured.

(1) Data Conditioning—The Input Manager "conditions" data to ensure inputs lie within predefined domains by dropping invalid data, conditioning data to be within range or marking data as out of range. *For example, physical signals can be limited in voltage, current, etc., while software data streams might include checks for not-a-number, protocol syntax compliance, etc. These are akin to syntactic data type checks.*

(2) Dynamic Consistency Checking—Ensures that data that passes a set of predefined checks and criteria to increase confidence to a level that the data is valid. *An example is the data from a magnetometer represented as a 32-bit unsigned integer. A 32-bit unsigned integer can contain a numeric range well beyond 0 to 360; range checking can be performed to validate acceptable values between 0 degrees to 360 degrees. Any values outside this range can be considered bad or invalid data and should be marked as invalid to not cause any unintended system behaviors. Another example is, if data is received on a bus from an independent upstream system, a risk of data corruption exists if no validity checking is performed. Current microcontrollers have built-in communication checks to help detect failures in protocols. One such criteria is to check for communication errors, that is, invalid message frames, incorrect parity, register overflows, and timely periodic rates.*

5.6.1.2 *Furthermore, multiple redundant data variables and their time histories may be considered together (for example, voting, estimation). Multiple low-quality sensors may be fused to provide higher data quality.*

5.6.1.3 The Input Manager may derive or estimate, or both, parameters from input data to be used by RTA components as additional "sanity check." These parameters can be used for dissimilar and redundant voting schemes providing additional data assurance.

5.6.2 *Requirements:*

5.6.2.1 *F3269-IM-001*—The IM shall accept as inputs, Assured External Data, Unassured External Data and, when monitoring the Complex Function behavior directly, Complex Function Outputs.

5.6.2.2 *F3269-IM-002*—The IM shall perform Data Conditioning on all unassured data.

5.6.2.3 *F3269-IM-003*—The IM shall perform Dynamic Consistency Checking on all data.

5.6.2.4 *F3269-IM-004*—The IM shall output Safety Monitor Inputs.

5.6.2.5 *F3269-IM-005*—The IM shall output Recovery Functions Inputs.

5.6.2.6 *F3269-IM-006*—The IM shall compute and output data quality attributes for all IM outputs, *such as update rate, accuracy, precision, and latency*.

5.7 *Safety Monitor:*

5.7.1 *Description:*

5.7.1.1 The Safety Monitor evaluates Larger System behavior by directly monitoring the Complex Function Output or Larger System behavior, or both. When directly monitoring Complex Function behavior, the Safety Monitor Inputs include Complex Function Output routed through the Input Manager. The Safety Monitor detects misbehaviors that indicate that the Larger System may leave the Safe State. This is accomplished by establishing Predefined Bounds to maintain the Larger System in a Safe State. A Safety Monitor Trigger Threshold is then defined to ensure that the Predefined Bounds are not violated by the RTA System. It has the goal of returning the RTA System to within the Safety Monitor Trigger Threshold, and eventually returning control to the Complex Function, if possible.

5.7.1.2 The Safety Monitor Coverage fully implements the defined RTA System Coverage. The Safety Monitor Function may consist of multiple monitoring functional capabilities. When multiple monitoring subfunctions are required, then a monitor arbitrator (such as a priority selector) is required.

5.7.1.3 Multiple monitoring subfunctions (Fig. 4) may be used to meet the necessary Monitor Coverage or achieve the required level of assurance, or both. The monitors may use functionally dissimilar methods and input parameters.

5.7.1.4 When more than one monitoring subfunction is implemented and potential exists for ambiguity in source at any given time, the overall RTA Safety Monitor, an Arbitrator function, is necessary to disambiguate conflicting monitor subfunction recommendations. This subfunction ensures that exactly one RTA Source Selection is made from the available RTA Output sources.

5.7.2 *Requirements:*

5.7.2.1 *F3269-SM-001*—The Safety Monitor shall accept Safety Monitor Inputs from the Input Manager.

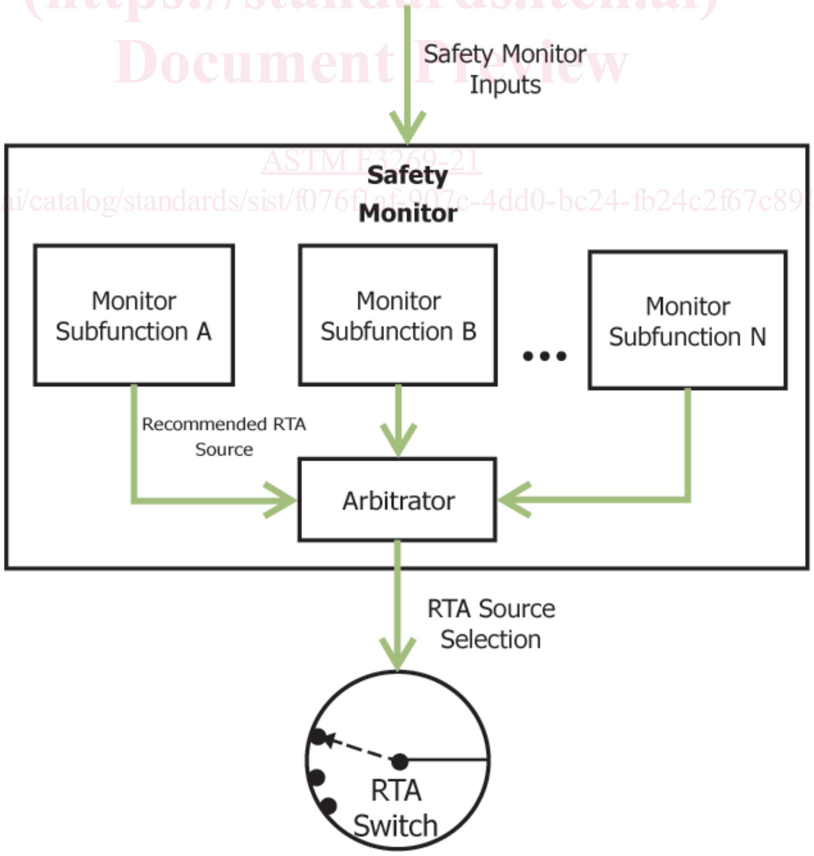

FIG. 4 Multiple Monitoring Subfunctions

5.7.2.2 *F3269-SM-002*—The Safety Monitor shall output exactly one *RTA Source Selection* to the RTA Switch at any point in time.

5.7.2.3 *F3269-SM-003*—The Safety Monitor shall continuously evaluate the Larger System behavior by monitoring:

(1) the Complex Function Output; or

(2) the RTA Output; or

(3) Larger System behavior; or

(4) any combination of the above.

5.7.2.4 *F3269-SM-004*—The Safety Monitor shall detect behaviors that indicate that the Larger System will leave the Safe State.

5.7.2.5 *F3269-SM-005*—When the Safety Monitor Trigger Threshold is exceeded, the Safety Monitor shall select and activate a suitable Recovery Function to ensure that the Predefined Bounds are not violated. *It has the goal of returning the RTA System to the Nominal Region, and eventually returning control to the Complex Function, if possible.*

5.7.2.6 *F3269-SM-006*—The Safety Monitor shall support the following transitions:

(1) Complex Function to Recovery Function.

(2) Recovery Function to a different Recovery Function.

(3) Recovery Function to the Complex Function.

5.7.2.7 *F3269-SM-007*—When more than one Recovery Function is available, the Safety Monitor shall have a priority scheme for selecting the appropriate Recovery Function. *A Recovery Function is considered available when the function is operational and applicable. A priority scheme could be to choose the Recovery Function with the lowest risk outcome.*

5.7.2.8 *F3269-SM-008*—When more than one Monitor Subfunction is implemented, an Arbitrator shall be implemented to select from potentially conflicting Monitor Subfunction recommendations.

5.7.2.9 *F3269-SM-009*—The Arbitrator shall evaluate each Monitor Subfunction recommendation and select the appropriate Recovery Function.

5.8 *RTA Switch:*

5.8.1 *Description:*

5.8.1.1 The RTA Switch (RS) is the means to switch the source of the RTA Output between functions (Complex Function, Recovery Function). Typically, the Complex Function is the source of the RTA Output, but the Safety Monitor can command an alternate source.

5.8.2 *Requirements:*

5.8.2.1 *F3269-RS-001*—The RS shall only accept RTA Source Selection from the Safety Monitor.

5.8.2.2 *F3269-RS-002*—The RS shall ensure that the only source of the RTA Output is the function (Complex Function or Recovery Function) selected by the Safety Monitor.

5.8.2.3 *F3269-RS-003*—The RS shall ensure that there is always exactly one source of RTA Output at all times.

5.9 *Recovery Function:*

5.9.1 *Description:*

5.9.1.1 A Recovery Function's purpose is to provide a known output that will maintain safe operation should the complex function misbehave. Recovery Function objectives are to maintain a Safe State and, when possible, return control to the Complex Function by re-entering the Nominal Region.

5.9.1.2 A Recovery Function provides an assured, alternative output to the Complex Function Output. When the Safety Monitor determines the RTA System has reached the Safety Monitor Trigger Threshold, the recovery function is selected. A Recovery Function is selectable by the Safety Monitor through the RTA Switch.

5.9.2 *Best Practices:*

5.9.2.1 *Re-enter Nominal Region*—The Recovery Function should be designed to return the RTA System to within the SMTT, to allow the Safety Monitor to reactivate the Complex Function. In some cases, re-entering the SMTT and reactivating the Complex Function may not be possible or desirable.

5.9.3 *Requirements:*

5.9.3.1 *F3269-RF-001*—The Recovery Function shall accept only assured data from the Input Manager.

5.9.3.2 *F3269-RF-002*—The Recovery Function shall ensure Recovery Function Output is available at time of selection.

5.9.3.3 *F3269-RF-003*—The Recovery Function should inform the Safety Monitor when the Recovery Function is unable to provide Recovery Function Output.

5.9.3.4 *F3269-RF-004*—The Recovery Function shall send Recovery Function Output to the RTA Switch.

5.9.3.5 *F3269-RF-005*—The Recovery Function shall maintain the RTA Output within the Predefined Bounds.

6. Keywords

6.1 adaptive; airworthiness; artificial intelligence; assurance; automated; autonomous software; autonomy; certification; complex; control systems; deep neural networks; fuzzy logic; machine learning; online verification; reinforcement learning; run-time assurance; safety; safety monitor; security; software; unmanned aircraft system; validation; verification

APPENDIXES

(Nonmandatory Information)

X1. GROUND COLLISION AVOIDANCE SYSTEM (GCAS) AS AN EXAMPLE RTA

INTRODUCTION

The intent of this appendix is to provide context for portions of this practice by discussing an example RTA system. The example system used here is a ground collision avoidance system (GCAS).

The functional purpose of a GCAS is to reduce the probability of controlled flight into terrain (CFIT) by avoiding ground impact while the aircraft is under controlled flight. The GCAS monitors aircraft state allowing the aircraft to be controlled by the pilot, ground control operator, or autonomous guidance system until ground impact is imminent. At that point, the GCAS takes control from the pilot and gives it to an autopilot to perform a recovery maneuver to avoid the ground. The GCAS returns control to the pilot when the aircraft velocity vector is clear of near terrain. The concept of operations for GCAS is to support flight at low altitude over terrain for both a piloted or unmanned aircraft.

An automatic GCAS has been fielded on U.S. Air Force F-16s (**1**).[6] GCAS has also been adapted to other aircraft that include general aviation fixed wing, as well as both small and large UAS (**2**). The examples and reference implementation provided do not reflect any one specific implementation. (See Fig. X1.1.)

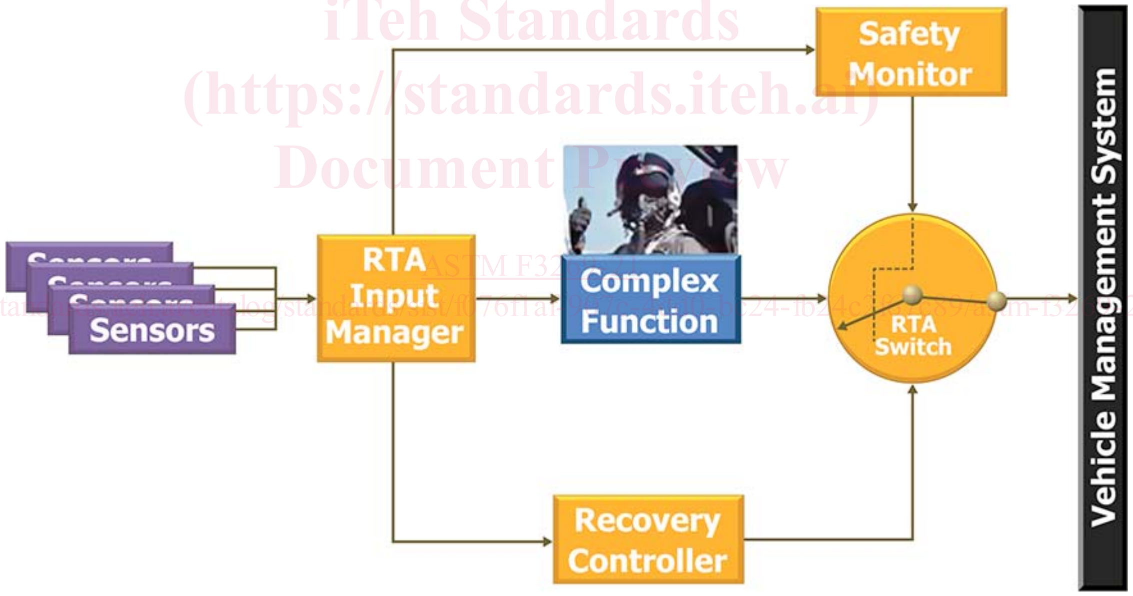

FIG. X1.1 Reference GCAS Implementation for Piloted Aircraft

[6] The boldface numbers in parentheses refer to the list of references at the end of this standard.

X1.1 Unassured Function

X1.1.1 The unassured function for GCAS is the pilot in the case of a piloted aircraft. The pilot, being an unassured function, controls the aircraft the vast majority of the time. Although the pilot is licensed,[7] on rare occasions the pilot may be disoriented, incapacitated, or otherwise incapable of reacting to imminent ground impact.

[7] While a pilot license is a form of pedigree, human performance includes behaviors and failure modes that are not fully addressed in the pedigree.