Annex A  Error Model Annex

Normative

Annex A.1  Error Model Annex Overview

(1) This annex defines an optional set of declarations and semantics to specify error models for components and connections in an AADL architecture specification. The language features defined in this annex enable specification of redundancy management and risk mitigation methods in an architecture, and enable qualitative and quantitative assessments of system properties such as safety, reliability, integrity, availability, and maintainability.

(2) This annex defines a sublanguage that can be used to declare error models within an error annex library and associate them with components in an architecture specification. An error model declares a set of error states for a component or connection, together with transitions and properties to specify how the error state of a component changes due to error events and error propagations. For example, the error state of a component might change due to an internal fault event, or due to an error propagated into that component from some other component. Alternatively, the error state of a component may be defined in terms of the error states of its subcomponents. For example, a component having internal redundancy might be in an erroneous state only when two or more of its subcomponents are in an erroneous state.

(3) The language features defined in this annex can be used to specify the risk mitigation methods employed in embedded computer system architectures to increase safety, reliability, integrity, and availability. Errors may propagate between components and connections, depending on the structure of the architecture specification. Language features are defined to specify how components vote to detect and mitigate errors in their subcomponents or the components on which they depend.

(4) From the bottom-up, the error models of low-level components typically capture the results of failure modes and effects analysis (e.g. as failure modes and effects analysis is defined in SAE ARP 4761). From the top-down, the error models of the overall system and high-level subsystems typically capture the results of system hazard analysis (e.g. as hazard analysis is defined in SAE ARP 4761). The rules defined in this annex assure that the results of these analyses as captured in an architecture specification are consistent and complete with respect to each other. For example, an architecture specification may be analyzed to determine if there are component failure effects that are not accounted for in any identified system hazard.

(5) The error behavior of a complete system emerges from the interactions between the individual component and connection error models. This annex defines the overall system error model as a composition of the error models of its components, where the composition depends on the structure and properties of the architecture specification. More formally, a component error model is a stochastic automaton, and the rules for composing component stochastic automata error models to form a system error model depend on the potential error propagations and error management behaviors declared in the architecture specification.

(6) The language features defined in this annex can be used for a variety of different purposes. This annex can support a number of the methods cited in SAE ARP4761, “Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment.” An architecture specification containing error models may be subjected to a variety of analysis methods. For example, fault trees can be generated from specifications to assess safety, Markov analyses can be applied to assess reliability and availability.
It is possible to check for consistency, completeness and traceability between the error models of interacting components, and between the error models of components and their subcomponents. This helps to ensure a globally consistent and complete error model for the overall architecture. For example, this enables an integrated approach that insures consistency and completeness between hazard analysis, failure modes and effects analysis, and the safety and reliability analyses that relate the two.

This annex defines language features for abstraction and mixed-fidelity modeling to help deal with the fact that models can become complex to understand and computationally challenging to analyze. An error model specification is divided into two parts, an error model type that defines error states that can be named in specifications of requirements and voting protocols, and one or more alternative error model implementations that may specify a more detailed model to use for quantitative analysis. The user may select whether a single simple error model is to be used as an abstraction for a given subsystem, or whether a detailed model should be constructed from the subcomponent and connection error models of that subsystem.

This annex supports a compositional approach to error modeling. This enables reuse of error models, makes it easier to modify architecture specifications and automatically regenerate safety and reliability models, facilitates abstraction and mixed-fidelity modeling, and enables improved traceability between architecture specifications and models and analysis results.

**Annex A.2 Referenced Documents**


**Annex A.3 Concepts and Terminology**

The definitions of this section are based on the concepts and terminology defined by IFIP WG10.4, slightly adapted to be compatible with the core AADL standard. The terms fault, fault latency, error, error latency, failure, and error propagation have the same meaning as defined in the core standard section 1.4 but are restated here in words that relate somewhat more clearly to other terms used in this annex.

A fault is an anomalous undesired change in the structure or data within a component that may cause that component to eventually fail to perform according to its nominal specification. Examples of faults are transistors burning out in hardware circuits, or programmers making coding mistakes when producing source text.
A fault places a component into an error state (possibly after some delay called the fault latency). An erroneous component may persist in that error state for some period of time before it behaves in a way that violates its nominal specification (called the error latency). For example, a burned out transistor in an adder circuit does not cause a processor to violate its nominal specification until that circuit is used and produces an incorrect output value.

A failure occurs when a component violates its nominal specification as a consequence of being in an error state. This is called a propagation of the error out of the component, which may occur with some error latency after the component first enters an erroneous state. For example, an erroneous component may send an incorrect value in a message or exceed its specified worst-case execution time.

A failure is the propagation of an error out of one component and into others. A failure may place the receiving components into error states, exactly like a fault. The distinction between a fault and a failure in this annex is that a fault is an internal or intrinsic event to a component, while a failure is an error propagation from the failed component into other components. The propagating component has failed, but the components into which the error propagates need not fail if they are tolerant to errors.

A component may also undergo a repair event. For example, the effects of a transient fault may disappear after some interval of time, or a failed component may be replaced. A repair event restores a component to an error-free state.

In this annex, both faults and repairs are declared as error events. An error event is any internal intrinsic event that changes the error state of a component. In this annex, an error state is simply a state that has meaning for error modeling purposes, it does not necessarily imply faulty behavior. For example, the initial error state is often assumed to be a fault-free state.

In this annex, failures are declared as error propagations. Each error propagation declared in an error model represents a type of failure effect that may affect the operation of other components. An error propagation combines the concept that a component is failing to meet its specification with the concept that a failure may cause another component to enter an error state. That is, it is assumed that a component fails only when another component might be affected by that failure and might consequently suffer an error itself. (An error propagation does not necessarily lead to an erroneous state in the dependent component because a dependent component may vote to mask error propagations.)

In the terminology of a failure modes and effects analysis, an error state is a failure mode, and an error propagation is a failure effect.

It is common practice to assume that the risk due to generic or design errors introduced by development-time faults (also known as design defects or bugs) has been reduced to an acceptable level prior to the start of system operation through the use of appropriate design assurance methods. For example, RTCA DO-178B and RTCA DO-254 provide guidelines for assuring that fielded avionics software code and hardware circuits have acceptably low risk of consequential design errors. If the features of this annex are nevertheless used to model the runtime effects of design defects, then an error event would typically be used to model the introduction of an error due to the run-time manifestation of a design defect. An error event would correspond to a flow of control and data that revealed a design defect, rather than to the development-time introduction of that design defect.