

The SAE Architecture Analysis & Design Language (AADL) Standard

Excerpt from the International Society for Automotive Engineers (SAE) Architecture Analysis & Design Language (AADL) Standard Document

www.aadl.info

info@aadl.info

Introduction

The SAE Architecture Analysis & Design Language (AADL) is a textual and graphical language used to design and analyze the software and hardware architecture of real-time systems and their performance-critical characteristics. The language is used to describe the structure of such systems as an assembly of software components mapped onto an execution platform. The language can describe functional interfaces to components (such as data inputs and outputs) and performance-critical aspects of components (such as timing). The language can describe how components interact, such as how data inputs and outputs are connected or how application software components are allocated to execution platform components. The language can also describe the dynamic behavior of the runtime architecture by supporting the modeling concept of operational modes and mode transitions. The language is designed to be extensible to accommodate analyses of the runtime architectures that the core language does not completely support. Extensions can take the form of new properties and analysis specific notations that can be associated with components.

The SAE AADL was developed to meet the special needs of performance-critical real-time systems, including embedded real-time systems such as avionics, automotive electronics, or robotics systems. The language can describe important performance-critical aspects such as timing requirements, fault and error behaviors, time and space partitioning, and safety and certification properties. Such a description allows a system designer to perform analyses of the composed components and systems such as system schedulability, sizing analysis, and safety analysis. From these analyses, the designer can evaluate architectural tradeoffs and changes. Since the AADL supports multiple and extensible analysis approaches, it provides the ability to analyze the cross cutting impacts of change in the architecture in one specification using multiple analysis tools. The AADL specification language has been designed to be further used with proper tool support to generate the code needed to integrate the system components and build a system executive. Since the models and the architecture specification drive the design and implementation, they can be maintained to permit model driven architecture based changes throughout the system lifecycle.

Scope

This standard defines a language for describing the software and execution platform architectures of performance-critical, embedded, real-time systems, known as the SAE Architecture Analysis & Design Language (AADL). An AADL architecture model describes the properties and interfaces of components such as application software modules and execution platform components such as processors, buses, and memory. It describes how these components interact and are

integrated to form complete systems. The AADL describes both functional interfaces and performance-critical aspects of individual components and assemblies of components. The runtime changes to the runtime architecture are modeled as operational modes and mode transitions. The language is applicable to real-time, resource-constrained, safety-critical systems that may include specialized device hardware.

This standard defines the core AADL that is designed to be extensible. While the core language provides a number of modeling concepts with precise semantics and with respect to mapping to execution platforms and execution, it is not possible to foresee all possible architecture analyses. Extensions to accommodate new analyses take the form of new properties and analysis specific notations that can be associated with components. Extension sets can be defined by users or vendors of tools. Extension sets can also be proposed for addition to the standard and will be defined in Annexes of this standard.

This standard does not specify how the internal design or implementation details of software and hardware components are to be specified. Those details can be specified by a variety of software programming and hardware description languages. The standard specifies relevant characteristics of external detailed design and implementation descriptions, such as source text written in a programming or hardware description language, as AADL component properties and rules of conformance between them.

This standard does not prescribe any particular system integration technologies, such as operating system or middleware application program interfaces or bus technologies or topologies. However, specific system architecture topologies, such as ARINC 653 RTOS, can be modeled through execution platform components. The AADL can be used to describe a variety of hardware architectures and a variety of software infrastructure and integration technologies can be used to implement a specified system. The standard specifies rules of conformance between AADL system architecture specifications and physical systems implemented from those specifications.

The standard was not designed around a particular set of tools. It is anticipated that systems and software tools will be provided to support the use of the AADL.

Purpose/Extent

The purpose of the AADL is to provide a standard and sufficiently precise (machine-processable) way of modeling the architecture of an embedded, real-time system, such as an avionics system or automotive control system to permit analysis of its performance-critical characteristics. Defining a standard way to describe system components, interfaces, and assemblies facilitates the exchange of engineering data between the multiple organizations and technical disciplines that are invariably involved in an embedded real-time system development effort. A precise and machine-processable way to describe conceptual and runtime architectures provides a framework for system modeling and analysis, facilitates the automation of several development activities, and significantly reduces design and implementation defects.

The AADL describes application software and execution platform components of a system, and the way in which components are assembled to form a complete system or subsystem. The language addresses the needs of system developers and can describe common functional (control and data flow) interfacing idioms as well as performance-critical aspects relating to timing, resource allocation, fault-tolerance, safety and certification.

The AADL describes functional interfaces and non-functional properties of application software and execution platform components. The language is not suited for internal design or implementation of components and is intended to be used in conjunction with existing standard languages in these areas. The AADL describes interfaces and properties of execution platform components including processor, memory, communication channels, and devices interfacing with

the external environment. Internal designs for such components may be specified by associating source text written in a hardware description language such as VHDL¹. The AADL can describe interfaces and properties of application software components implemented in source text, such as threads, processes, and runtime configurations. Internal designs and implementations for such components may be specified by associating source text written in a software programming language such as Ada95 or C, or domain-specific modeling languages such as MatLab[®]/Simulink[®].².

The AADL describes how components are composed together and how they interact to form complete system architectures. Runtime semantics of these components are specified in this standard. A variety of mechanisms is available to exchange control and data between components, including message passing, event passing, synchronized access to shared components, and remote subprogram calling. Thread scheduling protocols and timing requirements may be specified. Dynamic reconfiguration of the runtime architecture may be specified through operational modes and mode transitions. The language does not require the use of any specific hardware architecture or any specific runtime software infrastructure.

Rules of conformance are specified between specifications written in the AADL, source text and physical components described by those specifications, and physical systems constructed from those specifications. The AADL is not intended to describe all possible aspects of any possible component or system; selected syntactic and semantic requirements are imposed on components and systems. Compliance between AADL specifications and items described by specifications is determined through analysis, e.g., by tools for source text processing and system integration.

The AADL can be used for multiple activities in multiple development phases, beginning with preliminary system design. The language can be used by multiple tools to automate various levels of modeling, analysis, implementation, integration, verification and certification.

Field of Application

The AADL was developed to model embedded systems that have challenging resource (size, weight, power) constraints and strict real-time response requirements. Such systems should tolerate faults and utilize specialized hardware such as I/O devices. These systems are often certified to high levels of assurance. Intended fields of application include avionics systems, flight management, engine and power train control systems, certain medical devices, industrial process control equipment, robotics, and space applications. The AADL may be extended to support other applications as the need arises.

¹ VHDL is the "Very High Speed IC Hardware Description Language (Formerly Verilog Hardware Description Language). See IEEE VHDL Analysis and Standardization Group for details and status.

² MatLab and SimuLink are commercial tools available from The MathWorks.