# Multi-Dimensional Model Based Engineering for Performance Critical Computer Systems Using the AADL

B Lewis, P Feiler

# Multi-Dimensional Model Based Engineering for Performance Critical Computer Systems Using the AADL

B. Lewis[1], P Feiler[2]

1: US Army Research Development and Engineering Command, Software Engineering Directorate, Huntsville Al., USA 35898

2. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pa, USA 15213

**Abstract**: The Architecture Analysis & Design Language, (AADL), Society of Automotive Engineers (SAE), AS5506, was developed to support quantitative analysis of the runtime architecture of the embedded software system in computer systems with multiple critical operational properties, such as responsiveness, safety-criticality, security, and reliability by allowing a model of the system to be annotated with information relevant to each of these quality concerns and AADL to be extended with analysis-specific properties. It supports modelling of the embedded software runtime architecture, the computer system hardware, and the interface to the physical environment of embedded computer systems and system of systems. It was designed to support a full Model Based Engineering lifecycle including system specification, analysis, system tuning, integration, and upgrade by supporting modelling and analysis at multiple levels of fidelity. A system can be automatically integrated from AADL models when fully specified and when source code is provided for the software components.

**Keywords**: Architecture, Computer Architecture, Model Based Development, Architecture Design Language, Computer System Engineering, Computer Modelling, AADL, Architecture Analysis & Design Language.

## 1. Introduction

A key to engineering an embedded system is to analyze the operational characteristics of its realization as a software-intensive system through a model of its software runtime architecture, its compute platform, its interface to the physical environment, and the deployment of the software on the hardware platform. In order to achieve these operational characteristics, the architecture must provide valid data, at the right time, within resource limitations, with proper security, and provide required levels of fault tolerance and dependability.

Model-based engineering involves the creation of models of the system at a level of abstraction that is relevant to the analysis to be achieved. Traditionally, this has led to the creation of different models of the same system by different stakeholders. Dependability engineering resulted in the creation of fault trees for fault analysis and markov models for reliability analysis. Resource utilization analysis is based on resource demands and resource capacities. Scheduling analysis is based on a timing model of the application tasks. Security analysis involves the creation of a model in terms of security levels and domains applied to subjects and objects. In other words, a number of independently created models reflect the same system architecture and any change to this architecture during its life time requires each of these models to be updated and validated that it correctly reflects the actual system architecture. Similarly, it is difficult to consistently reflect any changes in one analysis dimension in other dimensions, e.g., consistently reflect the impact of choosing a different security encryption scheme on intrusion resistance as well as on schedulability and end-to-end latency.

So it becomes important to integrate the different analysis dimensions into a single architecture model. An architecture model that is annotated with analysis-specific information can drive model-based engineering by generating the analysis-specific models from this annotated model. This allows changes to the architecture to be reflected in the various analysis models with little effort by regenerating them from the architecture model This approach also allows us to evaluate the impact across multiple analysis dimensions.

The AADL was developed for just such a purpose. It was developed from significant experimentation and research over 15 years. It provides a language that is useful across domains where real-time, embedded, fault tolerant, secure, safety critical, software-intensive systems are developed. Its natural fields of application include avionics, automotive, autonomous systems, process control, and medical. Because of the strong need in today's complex systems, it has stimulated large industrial initatives to leverage the power of architectural multi-

dimensional analysis and automated system integration from the specifications and models.

## 2. Standard Development

### 2.1 History

The AADL language has been formed from three major areas. Its proof of concept and base for development was the MetaH language. MetaH was developed by Honeywell Labs, with Dr. Steve Vestal as principle investigator, over 12 years and three DARPA programs [1]. It was used in over 40 experimental projects, many of them DARPA programs, internal Honeywell investigations, Army experiments [2] and/or SEI experiments.

The second major area of input to the language was the other ADL languages developed by DARPA and within industry. Experience with these languages and MetaH, especially at the SEI by Dr. Peter Feiler, were also leveraged in the design of the AADL which broadened the domain of application and helped form the core AADL from which extensions would be later developed. Expertise on the standardization committee with UML and HOOD/STOOD were also leveraged to validate concepts, provide an industrial strength solution and ease integration within the industry.

The third major area of input was through the members of the SAE committee (see figure 1) which developed the requirements document, the core standard AS5506, and the annexes to the standard. Many language features were formed from the expressed needs of industry representatives from many of the major aviation and real time systems companies in the US and Europe. Many of the participants are leading engineers in their companies developing the next generation approaches for computer system development. These engineers recognized early the need for a common standard ADL to support computer system engineering in performance critical systems.

A fourth area now impacting the AADL is industrial pilots and university research programs that have adopted the AADL as a means to express performance critical architecture, analyze and then automate the integration of performance critical systems. These research programs along with industry experiences with AADL version 1.0 are providing and will provide guidance for AADL version 2.0, which will be balloted in summer, 2008.

Research projects in Europe have significantly contributed to the standard, and are expected to continue to do so. COTRE [3,4] and TOPCASED [5] are two Airbus led programs that have worked closely with the committee. The ASSERT [6] program, European Union funded and European Space Agency led, is also working closely with the committee. Several new research programs in the US and in Europe will be discussed later which will impact the standard into the future.

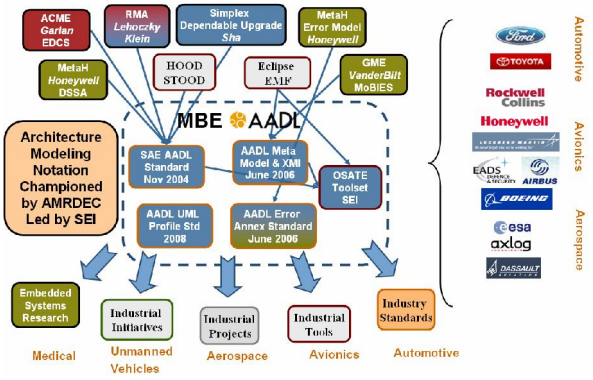


Figure 1 : Industry-Driven International SAE AADL Standard

### 2.2 Current State

The core AADL standard [7], version 1.0 was published in Nov. of 2004. A standard set of Annexes was published in June of 2006 [8].

These annexes included Graphical AADL Notation, AADL Meta-Model/XML Definition, Programming Language Guidelines, and Error Modeling.

Annexes provide a way of extending the language incrementally so that tools can support or use those aspects important for their domain of application. Several new annexes are under development.

The Behavior Annex has been prototyped and used on several research programs in Europe. It is now being updated for AADL version 2.0. It will have a strong fit with the core language.

The UML profile of AADL has been partially (as of this writing) implemented in the Rhapsody toolset based on a UML 2 meta-model developed by Ed Colbert. The OMG MARTE committee has developed and is prototyping guidelines for expressing AADL as a subset of the MARTE profile. As this subset is formalized into a full profile through a joint AADL/MARTE effort, it is expected to become the standard AADL profile for UML.

An annex is being developed to better support analysis in Net-Centric systems.

Based on the experience with SAE AADL in industrial projects improvements have been suggested. Version 2 of the AADL standard includes better support for architecture patterns, for modelling protocols, virtual channels, processor partitioning, and layered architectures. First ballot of AADL V2 is expected in Spring 2008.

### 2.3 AADL Language

The AADL provides components with well defined and precise semantics to describe computer system architecture. Precise semantics, hybrid automata,

are used to express system state and timing. Version 2 will add additional temporal logic expressions for ports and threads. Using the well defined and precise semantics allows quantitative analysis, system integration and glue code generation to build predictable systems. The language is designed to be incrementally refined as more information is available and provides permissions relative to the semantics.

Components have a type and one or more implementations. Software components include data, subprogram, thread, thread group and process. The hardware components include processor, memory, bus and device. The system component is used to describe hierarchical grouping of components, encapsulating software components, hardware components and lower level system components within their implementations.

Interfaces to components and component interactions are completely defined. The AADL supports data and event flow, synchronous call/return and shared access. In addition it supports end-to-end flow specifications that can be used to trace data or control flow through components.

The AADL supports real-time task scheduling using different scheduling protocols. Properties to support General Rate Monotonic Analysis and Earliest Deadline First are provided in the core standard.

The core also provides a property extension mechanism to define properties needed for additional forms of analysis. Execution semantics are defined for each category of component and specified in the standard with a hybrid automata notation.

Modal and configurable systems are supported by the AADL. Modes specify runtime transitions between statically known states and configurations of components, their connections and properties. Modes can be used for fault tolerant system reconfigurations affecting both hardware and software as well as software operational modes.

The AADL supports component evolution through inheritance, allowing more specific components to be refined from more abstract component. Large scale development is supported with packages which provide a name space and a library mechanism for components, as well as public and private sections. Packages support independent development and integration across contractors.

AADL language extensibility is supported both through a property sublanguage for specifying or modifying AADL properties. The AADL also provides an annex extension mechanism that can be used to specify sub-languages that will be processed within an AADL specification. An example is the Error Modeling Annex which allows specification of error models to be associated with core components. The combination of annex extensions and user defined property sets provide the means to integrate new specification capabilities and new analysis approaches to support the analysis tools and methods desired for multiple dimensions of analysis.

A number of other papers are referenced for a more detailed description of the language and are available on the AADL website [9] under "Home/Presentations and publications repository".

## 3. Model Based Engineering and Process Integration

### 3.1 Model-Based Engineering Process

The Model Based Engineering (MBE) for computer systems, as we use the term, includes the concepts of architectural specification, architectural analysis, and automated integration with generation of communication, glue code and the system executive to construct the final system. The MBE analyses are computer processable from the specification and quantitative. The AADL supports all the MBE concepts and adds significant additional capability and flexibility in a public standard. See figure 2 for the discussion below.

Architecture analysis is rerun each time the specification is updated, to provide model checking of the architecture as the architecture itself is refined (early trade-off analysis of architectural styles and hardware/communication effects or changes during the development or lifecycle) and as software components are developed and refined. Properties reflect the attributes of hardware and software components, connections, and ports and along with language semantics are used in analyses. Properties can capture estimates, requirements, or component options. Estimates may proceed to final values based on measurement as development proceeds. The integration of property values (estimates to measured) can be compared to higher level requirements.

Multiple architecture analysis methods, such as schedulability, latency, safety, are selected and run on the system model as it is incrementally developed. Models can be high level, low fidelity abstractions for some analyses, or early in development. The specification is refined during development and used to explore solutions as risks are discovered. Large models may be generated from system databases.

Analysis methods provide the cross checking needed to understand the effects of component or system change on schedulability, latency, safety, utilization, fault tolerance etc. The analyses themselves can be incrementally added as new analysis tools become available by adding the appropriate properties throughout the system lifecycle. System of systems integrators can collect AADL specifications of lower level systems and analyze the higher level integrated system. Analysis methods and analysis tools will have to be selected

consistent with the system architecture approach used. System developers will develop some special analysis for their own system implementation style and as a system engineering competitive advantage. Software component source code is supplied by the user and can be generated from component generators (such as from Matlab/Simulink or Beacon), hand coded, or reused/re-engineered. Given the AADL specification, source code for the application and the execution environment (processors, buses. memory, devices plus operating system, compilers etc), tools can automate the process of system configuration, composition and runtime system generation. These system integration/generation tools need to be consistent with the analysis methods and AADL semantics– generated according to the AADL specification. However, with generation, prototypes can be rapidly developed to experiment with the system effects due to architecture changes.

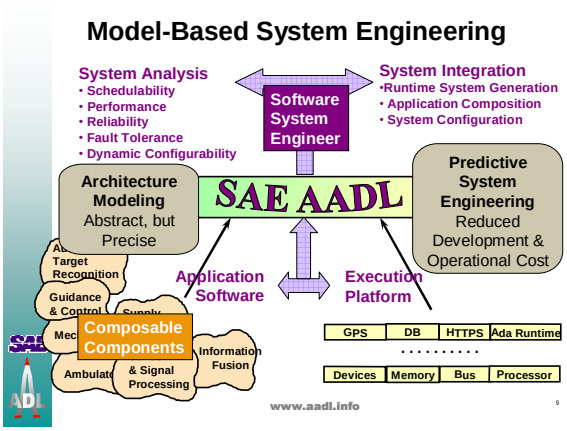


Figure 2 : Model Based System Engineering

An additional concept, the model bus, provides for automated, systematic integration of information from various engineering sources, tools and notations into the architecture specification. Advanced industrial research at Airbus and Boeing, as well as others, include the model bus for transferring data to and from models and to and from development and analysis tools.

Research at Rockwell has found that tools are typically extendable to allow more specific semantics and that the AADL makes an excellent intermediate language to represent the semantics of architecture in the databases and across tools. Transformations do not need to cover the whole domain of a specific language or toolset but only those parts that are needed to populate the analysis models. Model buses work from meta-models of the notations and transformations occur at the meta-model level.

## 4. Analysis Methods

Once the architecture or a critical subset of the architecture has been captured in an AADL specification, the properties can be added for the analysis of interest. For analysis, architecture data is extracted from the specification (by plug-ins in Eclipse based OSATE environment [9] and are solved within the plug-in or exported to an external analysis toolset. The ability to perform analyses can be incrementally added to a specification by adding properties, adding any additionally needed aspects of the architecture, and if the analysis interface does not already exist, tooling to extract from the specification the needed information to export. To develop the ability to run multiple analyses on the same architecture, properties are added for each analysis. See figure 3 for an example from the aviation domain, a partition specified as an AADL process with properties to support multiple analyses, in this case partition latency, security, and processor and memory utilization. This example also demonstrates the incremental capabilities built into the AADL, at this point the DisplayManager has only port groups defined in its interface, yet that is sufficient to evaluate data flow timing across partitions. This subsystem is part of a much larger example of incremental specification and multi-dimensional modelling in an avionics architecture made available [15].

AppSubSystems.aadl | AppSystems.aadl | errormodels.aadl

```
    SEI::RAMBudget => 350.0 KB;
end WarningAnnunciationManager;

process DisplayManager
  features
    DMToDisplay: port group AppTypes::PageReturn;
    DMToPCM: port group AppTypes::PageRequest;
  flows
    cmd_request: flow path DMToDisplay -> DMToPCM {
      Latency => 10 Ms;
      };
    show_page: flow path DMToPCM -> DMToDisplay {
      Latency => 50 Ms;
      };
  properties
    SEI::Partition_Latency => 50 Ms;
    SEI::Is_Partition => true;
    SEI::SecurityLevel => 3;
    SEI::MIPSBudget => 1200.0 MIPS;
    SEI::RAMBudget => 150.0 KB;
    SEI::ROMBudget => 50.0 KB;
end DisplayManager;
```

Figure 3 A Subsystem with Multiple Properties

The fact that the AADL can be used for multiple domains of analyses has been demonstrated as illustrated in Figure 4. At the center of the figure is the specification captured in the AADL. Multiple analysis methods from each of these domains have been demonstrated with AADL specifications using publicly available tools or internal tools. Analysis development has been a rapidly expanding area. A number of analyses not on this diagram have also

been demonstrated, such as simulation of AADL architectures, Petri net analysis, concurrency analysis based on process algebra and fault containment analysis.

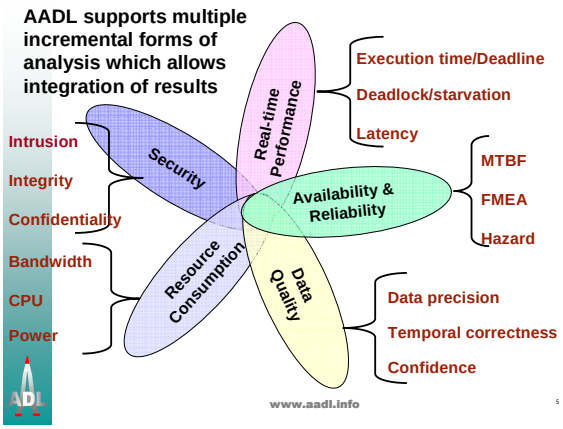


Figure 3 Analysis of Multiple Critical Qualities

Here are some example analyses. The latency analysis example provided by Rockwell [10] is based on an existing aviation architecture using the OSATE toolset and analysis plug-in. Its output indicates a violation in the real-time performance domain. 1600 flows were specified. See figure 4 below.

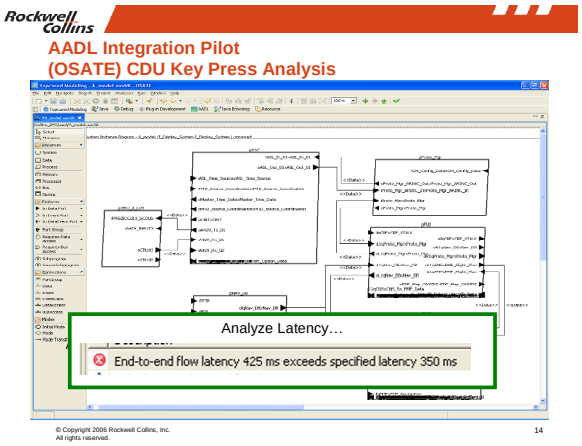


Figure 4 Example use of Latency Analysis

In the domains of data quality and security analysis, several analysis tools were added to OSATE during a recent research study at the SEI. This study was applying AADL and analysis techniques for the design of Sensor Networks but the analysis of security and data quality should be applicable to many applications.

Analysis in the resource consumption domain on a similar aviation system was demonstrated at Rockwell using internal analysis tools. The workload analysis results identified the possibility of reducing the number of processors, thereby decreasing system cost on each system. See Figure 5. Analysis of resource consumption using AADL is available through multiple toolsets. A partial list of tools is on the AADL website [9] under "Tool Integrators" page then "toolsets" and under "Home" page then "Rapid Growth Seen In New AADL Toolsets".

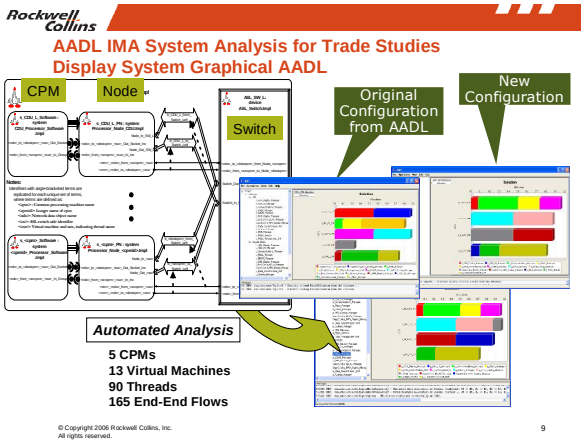


Figure 5 Aircraft Resource Consumption Modeling

In fault tolerant, safety critical systems error modeling is an important aspect of architectural design and should be integrated into the architecture specification so it can be cross checked against changes. The MetaH language originally supported reliability modeling and this capability has been significantly extended in the AADL through the Error Modelling annex to support multiple forms of safety and dependability analysis through error models that are attached to architectural components [11]. See Figure 6. Additional information on error modelling and plug-ins are also on the SEI website [9] under "Home" then "OSATE error modelling plug-in and other materials" for a guide to building error models.

Each component can have an error model. The architecture specification provides the foundation for understanding propagation and the effect of failure. Component error models integrate into a system error model through the architecture specification. Architecture specification changes then affect the error modeling directly making it much easier to maintain and discover impacts. See Figure 7.

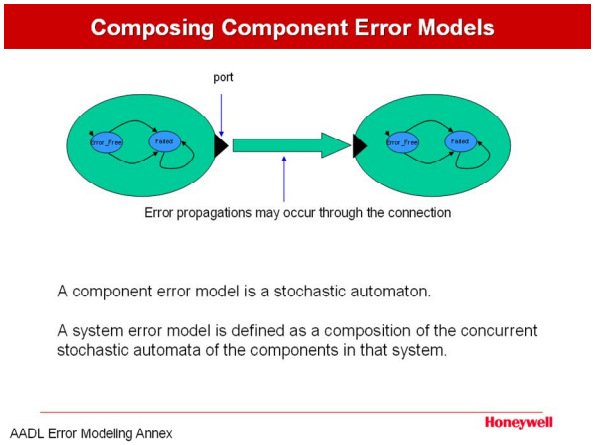


Figure 6 Component Error Models

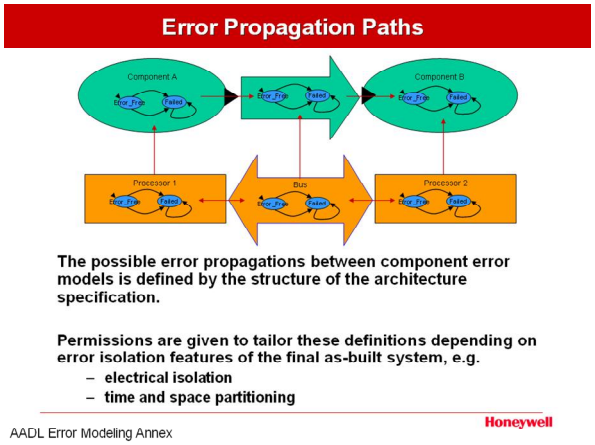


Figure 7 Architecture Integrates Error Models

Some of the supported analysis approaches include hazard analysis, failure modes and effects analysis (FMEA), fault trees, and Markov processes. Honeywell has demonstrated error modeling (hazard and FMEA) using annex capabilities on a large aircraft system [12]. ASSERT has modeled a dual redundant fault tolerant computer system using the annex [13,14].

Error models are used to develop multiple types of system dependability models and these can be analyzed within a single specification allowing consistency checking between them. See figure 8.

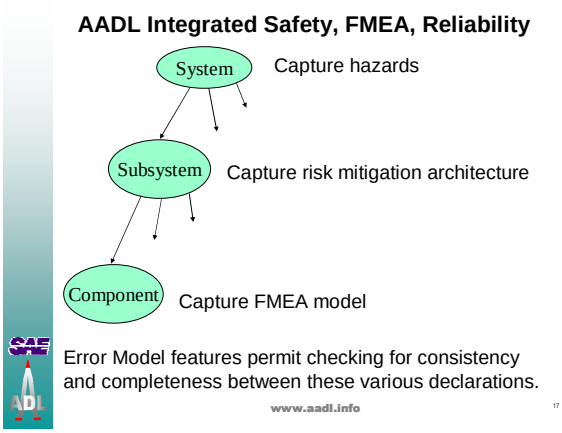


Figure 8 Integrating Dependability Analysis

Honeywell has done analysis on multiple aircraft platforms with a variety of buses, system synchronization approaches, and sizes. These analyses included resource consumption on processors and buses, global scheduling across multiple scheduling paradigms, switched network tuning, and FMEA and Hazard analysis [12]. These experiments and others among AADL users have had a direct impact on the AADL standard, demonstrating and refining capabilities for industrial application. See figure 9.

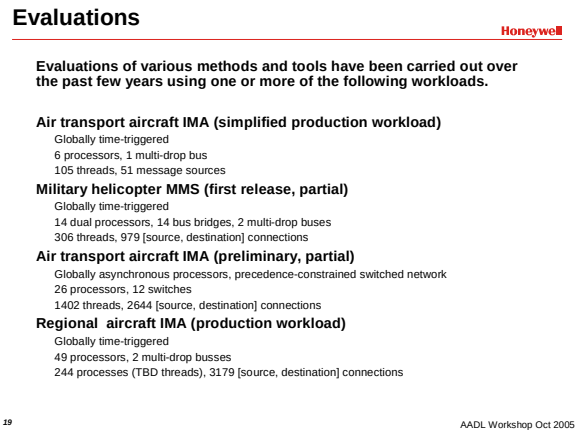


Figure 9 Aviation System Architectures Modeled

An aviation system AADL specification is available with instructions to demonstrate incremental, multi-fidelity analysis across multiple analysis dimensions. It has been posted on the SEI web site [15]. It uses the Open Source AADL Toolset Environment (OSATE) editor and analysis plug-ins. There are a number of AADL tools, many with analysis capability listed on the web site [9]. Specifications can be moved from tool to tool to leverage analysis capabilities. In many cases the tools are designed to work together but well defined semantics and a significant core capability have enabled

specifications to be moved from toolset to toolset to leverage analysis as demonstrated on ASSERT.

An important aspect in integrating multiple analysis approaches is the standardized AADL meta-model and XML schema. It provides a standard database format for analysis tool interaction. Tools can not only read from the XML models but also post back into it. For instance, a scheduling and binding toolset would be used to optimize processor and bus utilization given the constraints on binding captured in the AADL and the properties of threads and communication overheads. Given that binding, a reliability, latency or safety analysis could follow. Finally, from the system instance, the system could be automatically integrated with generation of glue code. See figure 10.

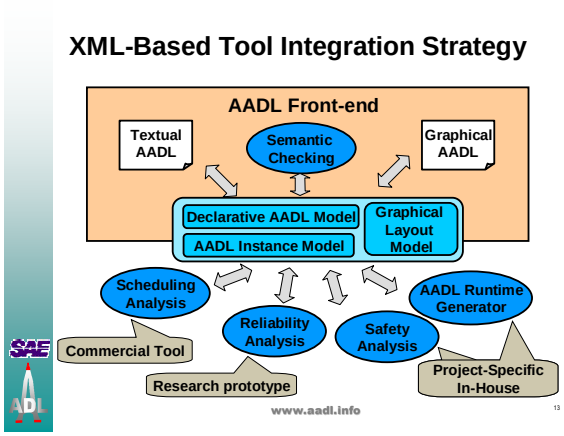


Figure 10 : Standardized and Simplified Tool Integration Supported

## 5. Significant Industrial Initiatives and Research Drive Future Capability

The AADL standard's capability and extensibility has attracted a number of industrial initiatives and research projects on embedded systems analysis and verification. Some of them are listed in figure 25. Most of these projects are cooperating with the AADL standardization committee to identify new capabilties needed in the language as well as developing analysis and code generation tools, and engineering methods. Industrial initiatives are piloting the technology to work out changes in processes to accommodate MBE using AADL, including acquisition, development, and certification. The ASSERT and the TOPCASED initiatives have already made significant contributions to the AADL language and tools. See figure 11.

The Support for Predictable Integration of mission Critical Embedded Systems (SPICES) project [16], as described in their presentation to the AADL committee, is strongly oriented to AADL modelling and analysis for use in containers which would be integrated to form systems. "The principal goals of SPICES are to extend the capacities of the microCCM component-based framework and to couple it with an AADL modeller in order to offer to system architects, software architects, and application designers a component-based modelling, design and analysis environment for distributed real-time embedded systems that should be deployed over heterogeneous targets such as GPP, DSP or FPGA." See Figure 12.

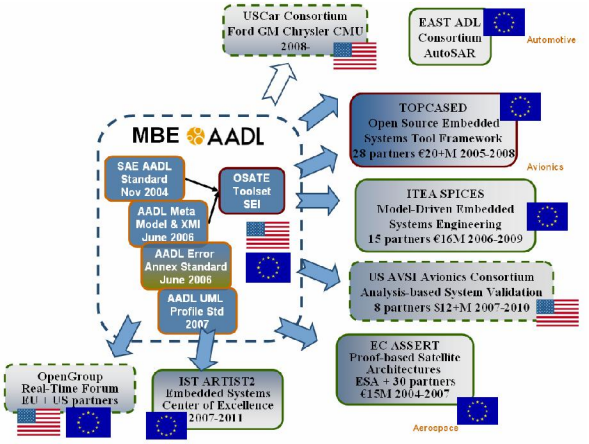


Figure 11 Industrial Initiatives Using the AADL

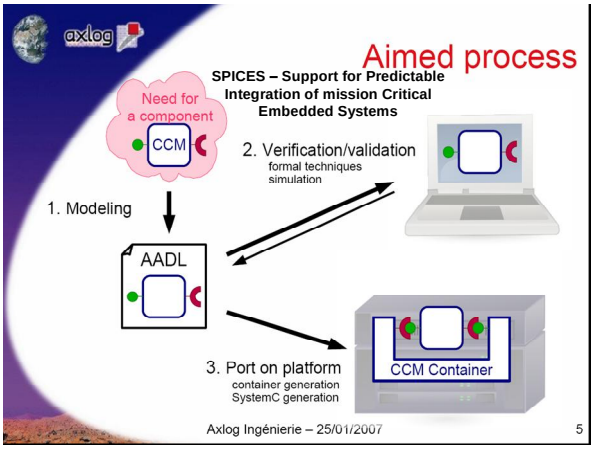


Figure 12 AADL modeling and Container Concept

Figure 13 provides an overview of three of the work packages. The first is to develop needed extensions to the AADL. These typically would be annexes like the Behaviour Annex Airbus is currently developing, but also corrections, extensions to the current standard where needed. The second work package is the development of additional AADL modelling tools and integrated verification and

validation tools. The third work package is the generation of validated FPGA's and validated software component containers and how to build tools to port them onto the platform. Two work packages not shown on the figure focus on pilots and dissemination.

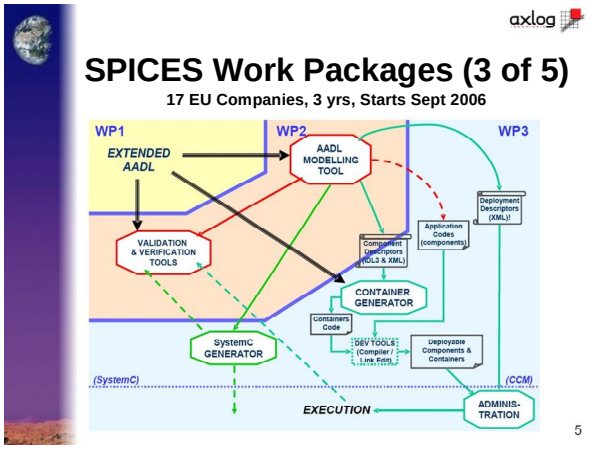


Figure 13 SPICES Extends AADL through System Building

Another new industrial initiative with a strong AADL focus is sponsored under the Aerospace Vehicle Systems Institute (AVSI) System and Software Integration Verification (SSIV) work tasks [17] and lead by Boeing. The AVSI is a cooperative of aerospace companies, DoD, and FAA to improve the integration of complex subsystems in aircraft. This AVSI project includes not only US contractors and system integrators but also European as well. Airbus and Dassault are joining with Boeing, Lockheed Martin and a number of US suppliers. There is an opportunity for European suppliers to also participate.

The theme of the AVSI-SSIV is "Integrate –then Build". It's objectives are listed in figure 14, and includes integrated analysis of the system, software, and hardware for performance, safety, security, and functionality. The prediction of system behaviour by analysis to assure acceptability occurs before the system building process begins through virtual integration. Some requirements are determined through analysis. These analyses will at least maintain existing levels of safety and security but are expected to improve them, since analysis and formal modelling will find issues testing and simulation often miss. This approach moves the industry beyond process toward evidence for safety and security.

As can be seen in figure 15, this approach modifies the acquisition process. The system integrator provides virtual models of "parts" to potential suppliers, who respond with their models of these subcomponents that fit the requirements they are bidding. Then these components are virtually re-integrated, with feedback, by the system integrator. This approach fits with investments made in Europe with the AADL and Model Based Engineering.

AVSI-SSIV is designed as a multi-phased program. The first phase was a planning phase and is near completion. The next phase is a one year task to demonstrate viability of the approach, which includes demonstrating AADL, analysis and the model bus and develop a new model based acquisition process for industry. This next phase will start in the spring of 2008 and is estimated at about $4M. Additional work packages are being defined for follow on research and industrial process definition, which if funded, could reach $40M.


Figure 14 AVSI-SSIV Objectives

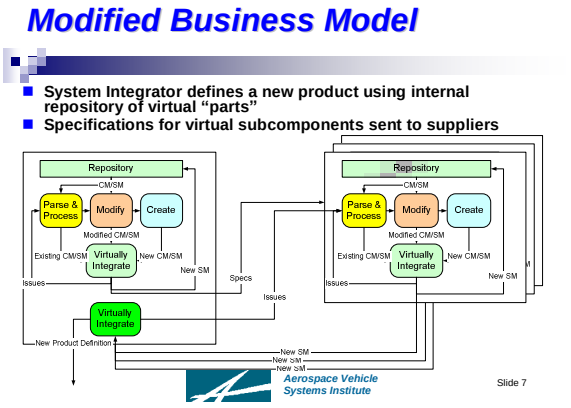


Figure 15 AVSI-SSIV Virtual Integration Based Acquisition

Figure 16 illustrates the way the model bus will be used to integrate models, exchange information

between tools and between the system integrator and the suppliers. The model bus plays an important role by keeping the approach from depending on specific tools. It allows the integration of tool results into the models and the models with analysis tools. The AADL was selected after an extensive study of architecture description languages in industry and academia for its capability, existing tools, and extensibility mechanisms.

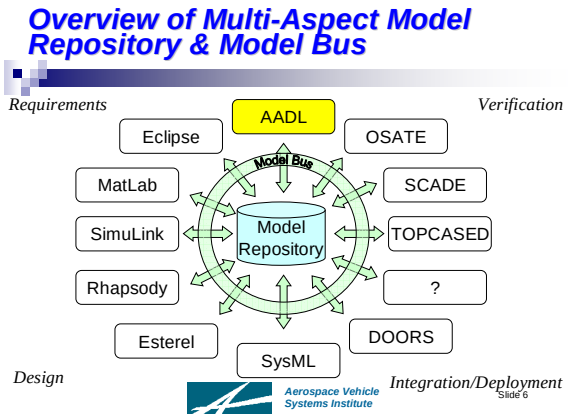


Figure 16 Integrated Tools/System Model via Model Bus & Repository

Both the AVSI-SSIV and the SPICES programs are interested in exploring incremental certification approaches and the impact that quantitative/formal modelling can have on certification and testing.

## 6. Conclusion

The AADL's standardized component based, well defined, semantics for runtime system architecture, integrating the computer hardware and software specifically supports capturing the architecture for multi-dimension analysis. As such, it becomes an excellent place to integrate computer system engineering requirements, platform development and software engineering concerns. New research projects are likely to add new analyses and more highly automated approaches for system integration as well as new industrial acquisition processes which leverage model based engineering across multiple critical system performance dimensions.

The AADL provides or supports through tools significant model based embedded system engineering benefits. These include:

- Precise and well defined semantics supporting analyzable models to predict system performance and drive development
- Prediction of system runtime characteristics at different fidelity
- Bridge between application engineer, architect and software engineer
- Prediction early and throughout lifecycle
- Reduced integration and maintenance effort

The AADL also provides additional benefit based on its standardized features including:

- Common modelling notation across organizations
- Single architecture model augmented with properties
- Interchange & integration of architecture models
- Tool interoperability & integrated engineering environments

## 7. Acknowledgement

The author would like to thank the AADL standardization committee for their diligent efforts and willingness to share through their presentations what they are doing with the AADL. Many of the papers referenced below have been developed by committee members.

## 8. References

[1] Binns, Matt Englehart, Mike Jackson and Steve Vestal, "Domain Specific Software Architectures for Guidance, Navigation and Control," Honeywell Technology Center, Minneapolis, MN, International Journal of Software Engineering and Knowledge Engineering, Vol6, No. 2, 1996, pages 201-227.

[2] David J. McConnell, Bruce Lewis and Lisa Gray, "Reengineering a Single Threaded Embedded Missile application onto a Parallel Processing Platform using MetaH," 5t Workshop on Parallel and Distributed Real Time Systems, 1996.

[3] Patrick Farail, Pierre Dissaux, "COTRE a Software Design Workshop", DASIA 2002, May 2002.

[4] Pierre Gaufillet, « COTRE as an AADL profile », AADL Standardization Meeting, Edinburgh, July 2004

[5] Pierre Gaufillet, « TOPCASED – Toolkit In Open source for Critical Applications & SystEms Development », AADL Workshop, Paris, Oct 2005, http://www.topcased.org/

[6] Eric Conquet, « ASSERT – Automated proof-based System and Software Engineering for

Real-Time systems », AADL Standardization Meeting, Edinburgh, July 2004.

[7] Society of Automotive Engineers (SAE) Avionics Systems Division (ASD) AS-2C Subcommittee. “Avionics Architecture Description Language Standard.”, AS 5506, November 2004

[8] "Society of Automotive Engineers (SAE) Avionics Systems Division (ASD) AS-2C Subcommittee. “SAE Architecture Analysis & Design Language (AADL) Annex Volume 1 : Graphical AADL Notation, AADL Meta-Model and Interchange Formats, Language Compliance and Application Program Interface"; 2006 June. Report No.: AS 5506 /1.

[9] AADL Documents, OSATE Toolset, Error Modeling Plug-ins, etc, www.aadl.info

[10] Mettenburg J. "Industrial Applications of AADL - AADL Avionics Case Studies and Concepts for integrating AADL into System Development". In: AADL Standardization Meeting; 2007 January; 2007.

[11] Vestal S. "An Overview of the Architecture Analysis & Design Language (AADL) Error Model Annex ". In: AADL Workshop; 2005 October; Paris; 2005

[12] Steve Vestal, « Automating Timing and Safety Analyses from Architecture Specifications », AADL Standardization Meeting, April 2005

[13] Ana Rugina, Karama Kanoun, Mohamed Kaaniche, Jeremie Guiochet, « Dependability modelling of a fault tolerant duplex system using AADL and GSPNs », LAAS Research Report no. 05315, Draft of ASSERT Report, Sept 2005.

[14] Ana Rugina, « Dependability Modelling using AADL and the AADL Error Model Annex », AADL Workshop, Paris, Oct 2005

[15] Peter Feiler, “Evolution of an Avionics System”, Paper at www.aadl.info/downloads/Models/Evolution , AADL models at www.aadl.info/downloads/Models/IntegratedModel10292007.zip

[16] Tilman J-F, Sezestre R, Schyn A. "SPICES – Support for Predictable Integration of mission Critical Embedded Systems". In: AADL Standardization Meeting; 2007 January; 2007.

[17] John Chilenski, “Aerospace Vehicle Systems Institute Systems and Software Integration Verification Overview”, Nov 27, 2007, AADL Safety and Security Modeling Meeting.

## 9. Glossary

AADL: Architecture Analysis & Design Language
ADL: Architecture Design Language
ASSERT: Automated proof-based System and Software Engineering for Real-Time systems
OSATE: Open Source AADL Toolset Environment
TOPCASED: Toolkit In Open source for Critical Applications & SystEms Development

# Multi-Dimensional Model Based Engineering for Performance Critical Computer Systems Using the AADL

B. Lewis[1], P Feiler[2]

1: US Army Research Development and Engineering Command, Software Engineering Directorate, Huntsville Al., USA 35898

2. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pa, USA 15213

**Abstract**: The Architecture Analysis & Design Language, (AADL), Society of Automotive Engineers (SAE), AS5506, was developed to support quantitative analysis of the runtime architecture of the embedded software system in computer systems with multiple critical operational properties, such as responsiveness, safety-criticality, security, and reliability by allowing a model of the system to be annotated with information relevant to each of these quality concerns and AADL to be extended with analysis-specific properties. It supports modelling of the embedded software runtime architecture, the computer system hardware, and the interface to the physical environment of embedded computer systems and system of systems. It was designed to support a full Model Based Engineering lifecycle including system specification, analysis, system tuning, integration, and upgrade by supporting modelling and analysis at multiple levels of fidelity. A system can be automatically integrated from AADL models when fully specified and when source code is provided for the software components.

**Keywords**: Architecture, Computer Architecture, Model Based Development, Architecture Design Language, Computer System Engineering, Computer Modelling, AADL, Architecture Analysis & Design Language.

## 1. Introduction

A key to engineering an embedded system is to analyze the operational characteristics of its realization as a software-intensive system through a model of its software runtime architecture, its compute platform, its interface to the physical environment, and the deployment of the software on the hardware platform. In order to achieve these operational characteristics, the architecture must provide valid data, at the right time, within resource limitations, with proper security, and provide required levels of fault tolerance and dependability.

Model-based engineering involves the creation of models of the system at a level of abstraction that is relevant to the analysis to be achieved. Traditionally, this has led to the creation of different models of the same system by different stakeholders. Dependability engineering resulted in the creation of fault trees for fault analysis and markov models for reliability analysis. Resource utilization analysis is based on resource demands and resource capacities. Scheduling analysis is based on a timing model of the application tasks. Security analysis involves the creation of a model in terms of security levels and domains applied to subjects and objects. In other words, a number of independently created models reflect the same system architecture and any change to this architecture during its life time requires each of these models to be updated and validated that it correctly reflects the actual system architecture. Similarly, it is difficult to consistently reflect any changes in one analysis dimension in other dimensions, e.g., consistently reflect the impact of choosing a different security encryption scheme on intrusion resistance as well as on schedulability and end-to-end latency.

So it becomes important to integrate the different analysis dimensions into a single architecture model. An architecture model that is annotated with analysis-specific information can drive model-based engineering by generating the analysis-specific models from this annotated model. This allows changes to the architecture to be reflected in the various analysis models with little effort by regenerating them from the architecture model. This approach also allows us to evaluate the impact across multiple analysis dimensions.

The AADL was developed for just such a purpose. It was developed from significant experimentation and research over 15 years. It provides a language that is useful across domains where real-time, embedded, fault tolerant, secure, safety critical, software-intensive systems are developed. Its natural fields of application include avionics, automotive, autonomous systems, process control, and medical. Because of the strong need in today's

complex systems, it has stimulated large industrial initatives to leverage the power of architectural multi-dimensional analysis and automated system integration from the specifications and models.

## 2. Standard Development

### 2.1 History

The AADL language has been formed from three major areas. Its proof of concept and base for development was the MetaH language. MetaH was developed by Honeywell Labs, with Dr. Steve Vestal as principle investigator, over 12 years and three DARPA programs [1]. It was used in over 40 experimental projects, many of them DARPA programs, internal Honeywell investigations, Army experiments [2] and/or SEI experiments.

The second major area of input to the language was the other ADL languages developed by DARPA and within industry. Experience with these languages and MetaH, especially at the SEI by Dr. Peter Feiler, were also leveraged in the design of the AADL which broadened the domain of application and helped form the core AADL from which extensions would be later developed. Expertise on the standardization committee with UML and HOOD/STOOD were also leveraged to validate concepts, provide an industrial strength solution and ease integration within the industry.

The third major area of input was through the members of the SAE committee (see figure 1) which developed the requirements document, the core standard AS5506, and the annexes to the standard. Many language features were formed from the expressed needs of industry representatives from many of the major aviation and real time systems companies in the US and Europe. Many of the participants are leading engineers in their companies developing the next generation approaches for computer system development. These engineers recognized early the need for a common standard ADL to support computer system engineering in performance critical systems.

A fourth area now impacting the AADL is industrial pilots and university research programs that have adopted the AADL as a means to express performance critical architecture, analyze and then automate the integration of performance critical systems. These research programs along with industry experiences with AADL version 1.0 are providing and will provide guidance for AADL version 2.0, which will be balloted in summer, 2008.

Research projects in Europe have significantly contributed to the standard, and are expected to continue to do so. COTRE [3,4] and TOPCASED [5] are two Airbus led programs that have worked closely with the committee. The ASSERT [6] program, European Union funded and European Space Agency led, is also working closely with the committee. Several new research programs in the US and in Europe will be discussed later which will impact the standard into the future.

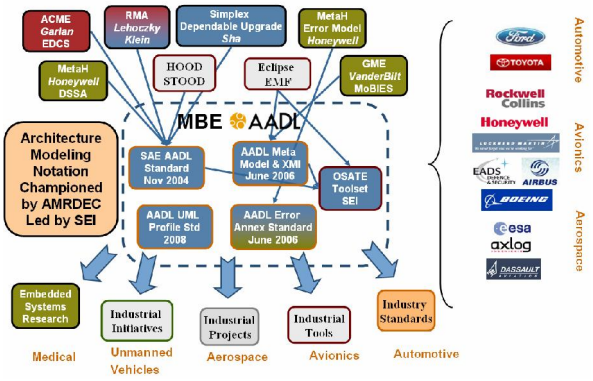


Figure 1 : Industry-Driven International SAE AADL Standard

### 2.2 Current State

The core AADL standard [7], version 1.0 was published in Nov. of 2004. A standard set of Annexes was published in June of 2006 [8]. These annexes included Graphical AADL Notation, AADL Meta-Model/XML Definition, Programming Language Guidelines, and Error Modelling.

Annexes provide a way of extending the language incrementally so that tools can support or use those aspects important for their domain of application. Several new annexes are under development. The Behaviour Annex has been prototyped and used on several research programs in Europe. It is now being updated for AADL version 2.0. It will have a strong fit with the core language.

The UML profile of AADL has been partially (as of this writing) implemented in the Rhapsody toolset based on a UML 2 meta-model developed by Ed Colbert. The OMG MARTE committee has developed and is prototyping guidelines for expressing AADL as a subset of the MARTE profile. As this subset is formalized into a full profile through a joint AADL/MARTE effort, it is expected to become the standard AADL profile for UML. Also, an annex is being developed to better support analysis in Net-Centric systems.

Based on experience with SAE AADL in industrial projects, improvements to the language have been suggested. Version 2 of the AADL standard includes better support for architecture patterns, for modelling

protocols, virtual channels, processor partitioning, and layered architectures. First ballot of AADL V2 is expected in Spring 2008.

### 2.3 AADL Language

The AADL provides components with well defined and precise semantics to describe computer system architecture. Precise semantics, hybrid automata, are used to express system state and timing. Version 2 will add additional temporal logic expressions for ports and threads. Using the well defined and precise semantics allows quantitative analysis, system integration and glue code generation to build predictable systems. The language is designed to be incrementally refined as more information is available and provides permissions relative to the semantics.

Components have a type and one or more implementations. Software components include data, subprogram, thread, thread group and process. The hardware components include processor, memory, bus and device. The system component is used to describe hierarchical grouping of components, encapsulating software components, hardware components and lower level system components within their implementations.

Interfaces to components and component interactions are completely defined. The AADL supports data and event flow, synchronous call/return and shared access. In addition it supports end-to-end flow specifications that can be used to trace data or control flow through components.

The AADL supports real-time task scheduling using different scheduling protocols. Properties to support General Rate Monotonic Analysis and Earliest Deadline First are provided in the core standard.
The core also provides a property extension mechanism to define properties needed for additional forms of analysis. Execution semantics are defined for each category of component and specified in the standard with a hybrid automata notation.

Modal and configurable systems are supported by the AADL. Modes specify runtime transitions between statically known states and configurations of components, their connections and properties. Modes can be used for fault tolerant system reconfigurations affecting both hardware and software as well as software operational modes.

The AADL supports component evolution through inheritance, allowing more specific components to be refined from more abstract component. Large scale development is supported with packages which provide a name space and a library mechanism for components, as well as public and private sections. Packages support independent development and integration across contractors.

AADL language extensibility is supported both through a property sublanguage for specifying or modifying AADL properties. The AADL also provides an annex extension mechanism that can be used to specify sub-languages that will be processed within an AADL specification. An example is the Error Modeling Annex which allows specification of error models to be associated with core components. The combination of annex extensions and user defined property sets provide the means to integrate new specification capabilities and new analysis approaches to support the analysis tools and methods desired for multiple dimensions of analysis.
A number of other papers are referenced for a more detailed description of the language and are available on the AADL website [9] under "Home/Presentations and publications repository".

## 3. Model Based Engineering and Process Integration

### 3.1 Model-Based Engineering Process

The Model Based Engineering (MBE) for computer systems, as we use the term, includes the concepts of architectural specification, architectural analysis, and automated integration with generation of communication, glue code and the system executive to construct the final system. The MBE analyses are computer processable from the specification and quantitative. The AADL supports all the MBE concepts and adds significant additional capability and flexibility in a public standard. See figure 2 for the discussion below.

Architecture analysis is rerun each time the specification is updated, to provide model checking of the architecture as the architecture itself is refined (early trade-off analysis of architectural styles and hardware/communication effects or changes during the development or lifecycle) and as software components are developed and refined. Properties reflect the attributes of hardware and software components, connections, and ports and along with language semantics are used in analyses. Properties can capture estimates, requirements, or component options. Estimates may proceed to final values based on measurement as development proceeds. The integration of property values (estimates to measured) can be compared to higher level requirements.

Multiple architecture analysis methods, such as schedulability, latency, safety, are selected and run

on the system model as it is incrementally developed. Models can be high level, low fidelity abstractions for some analyses, or early in development. The specification is refined during development and used to explore solutions as risks are discovered. Large models may be generated from system databases.

Analysis methods provide the cross checking needed to understand the effects of component or system change on schedulability, latency, safety, utilization, fault tolerance etc. The analyses themselves can be incrementally added as new analysis tools become available by adding the appropriate properties throughout the system lifecycle. System of systems integrators can collect AADL specifications of lower level systems and analyze the higher level integrated system. Analysis methods and analysis tools will have to be selected consistent with the system architecture approach used. System developers will develop some special analysis for their own system implementation style and as a system engineering competitive advantage. Software component source code is supplied by the user and can be generated from component generators (such as from Matlab/Simulink or Beacon), hand coded, or reused/re-engineered. Given the AADL specification, source code for the application and the execution environment (processors, buses. memory, devices plus operating system, compilers etc), tools can automate the process of system configuration, composition and runtime system generation. These system integration/generation tools need to be consistent with the analysis methods and AADL semantics–generated according to the AADL specification. However, with generation, prototypes can be rapidly developed to experiment with the system effects due to architecture changes.

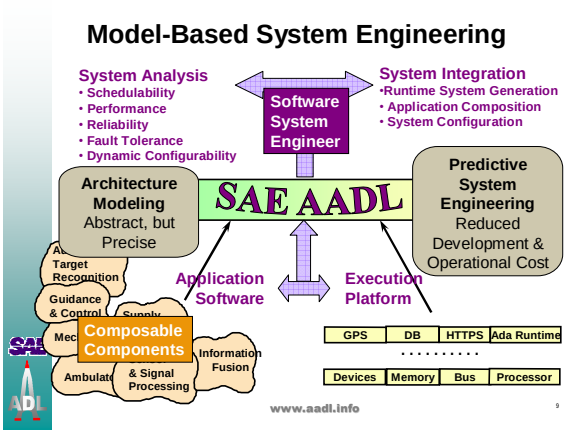


Figure 2 : Model Based System Engineering

An additional concept, the model bus, provides for automated, systematic integration of information from various engineering sources, tools and notations into the architecture specification. Advanced industrial research at Airbus and Boeing, as well as others, include the model bus for transferring data to and from models and to and from development and analysis tools.

Research at Rockwell has found that tools are typically extendable to allow more specific semantics and that the AADL makes an excellent intermediate language to represent the semantics of architecture in the databases and across tools. Transformations do not need to cover the whole domain of a specific language or toolset but only those parts that are needed to populate the analysis models. Model buses use meta-model specifications of the notations and transformations occur at the meta-model level.

## 4. Analysis Methods

Once the architecture or a critical subset of the architecture has been captured in an AADL specification, the properties can be added for the analysis of interest. For analysis, architecture data is extracted from the specification (by plug-ins in Eclipse based OSATE environment [9] and are solved within the plug-in or exported to an external analysis toolset. The ability to perform analyses can be incrementally added to a specification by adding properties, adding any additionally needed aspects of the architecture, and if the analysis interface does not already exist, plug-ins to extract from the specification the needed information to export. To develop the ability to run multiple analyses on the same architecture, properties are added for each analysis. See figure 3 for an example from the aviation domain, a partition specified as an AADL process with properties to support multiple analyses, in this case partition latency, security, and processor and memory utilization. This example also demonstrates the incremental capabilities built into the AADL, at this point the DisplayManager has only port groups defined in its interface, yet that is sufficient to evaluate data flow timing across partitions. This subsystem is part of a much larger example of incremental specification and multi-dimensional modelling in an avionics architecture made available [15].

```
AppSubSystems.aadl | AppSystems.aadl | errormodels.aadl
      SEI::RAMBudget => 350.0 KB;
end WarningAnnunciationManager;

process DisplayManager
  features
    DMToDisplay: port group AppTypes::PageReturn;
    DMToPCM: port group AppTypes::PageRequest;
  flows
    cmd_request: flow path DMToDisplay -> DMToPCM {
      Latency => 10 Ms;
      };
    show_page: flow path DMToPCM -> DMToDisplay {
      Latency => 50 Ms;
      };
  properties
    SEI::Partition_Latency => 50 Ms;
    SEI::Is_Partition => true;
    SEI::SecurityLevel => 3;
    SEI::MIPSBudget => 1200.0 MIPS;
    SEI::RAMBudget => 150.0 KB;
    SEI::ROMBudget => 50.0 KB;
end DisplayManager;
```

Figure 3 A Subsystem with Multiple Properties

The fact that the AADL can be used for multiple domains of analyses has been demonstrated as illustrated in Figure 4. At the center of the figure is the specification captured in the AADL. Multiple analysis methods from each of these domains have been demonstrated with AADL specifications using publicly available tools or company internal tools. Analysis development has been a rapidly expanding area with the expressability of the AADL. A number of analyses not on this diagram have also been demonstrated, such as simulation of AADL architectures, Petri net analysis, concurrency analysis based on process algebra and fault containment analysis.

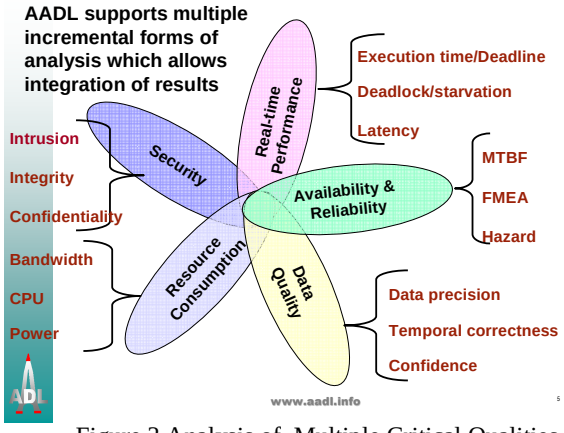


Figure 3 Analysis of Multiple Critical Qualities

Here are some example analyses. The latency analysis example provided by Rockwell [10] is based on an existing aviation architecture using the OSATE toolset and analysis plug-in. Its output indicates a violation in the real-time performance domain. 1600 flows were specified. See figure 4 below.

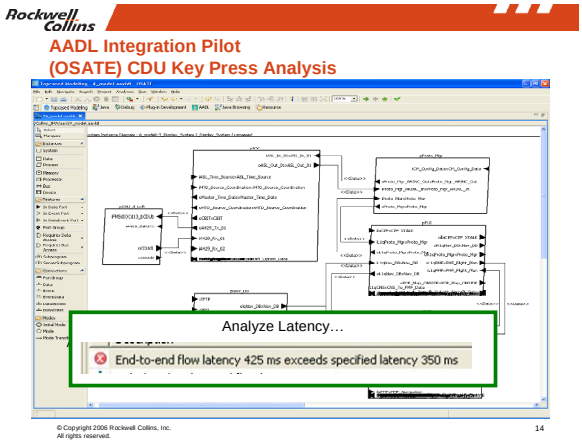


Figure 4 Example use of Latency Analysis

In the domains of data quality and security analysis, several analysis tools were added to OSATE during a recent research study at the SEI. This study was applying AADL and analysis techniques for the design of Sensor Networks but the analysis of security and data quality should be applicable to many applications.

Analysis in the resource consumption domain on a similar aviation system was demonstrated at Rockwell using internal analysis tools. The workload analysis results identified the possibility of reducing the number of processors, thereby decreasing system cost on each system. See Figure 5. Analysis of resource consumption using AADL is available through multiple toolsets. A partial list of tools is on the AADL website [9] under "Tool Integrators" page then "toolsets" and under "Home" page then "Rapid Growth Seen In New AADL Toolsets".

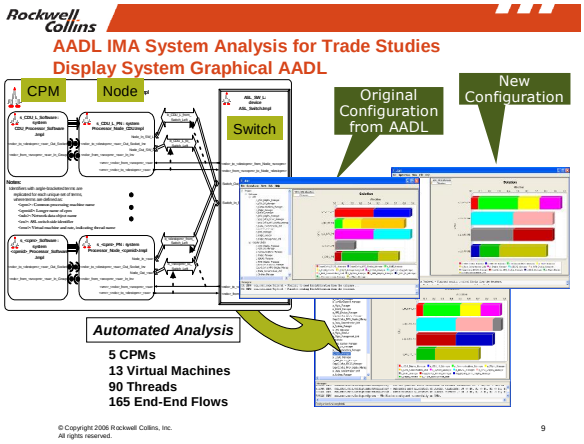


Figure 5 Aircraft Resource Consumption Modeling

In fault tolerant, safety critical systems error modeling is an important aspect of architectural design and should be integrated into the architecture

specification so it can be cross checked against changes. The MetaH language originally supported reliability modeling and this capability has been significantly extended in the AADL through the Error Modelling annex to support multiple forms of safety and dependability analysis through error models that are attached to architectural components [11]. See Figure 6. Additional information on error modelling and plug-ins is also on the SEI website [9]. See under "Home" then "OSATE error modelling plug-in and other materials" for a guide to building error models.

Each component can have an error model. The architecture specification provides the foundation for understanding propagation and the effect of failure. Component error models integrate into a system error model through the architecture specification. Architecture specification changes then affect the error modeling directly making it much easier to maintain and discover impacts. See Figure 7.

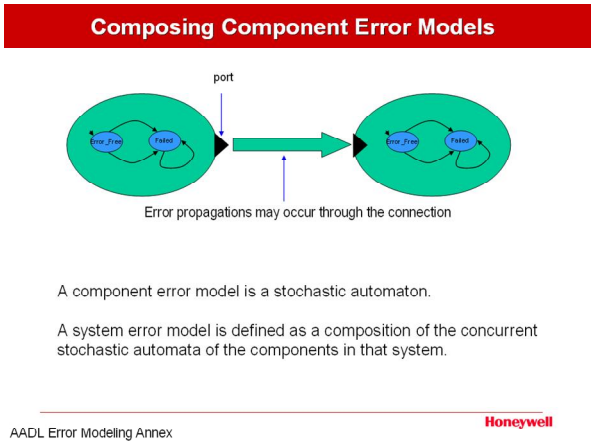


Figure 6 Component Error Models

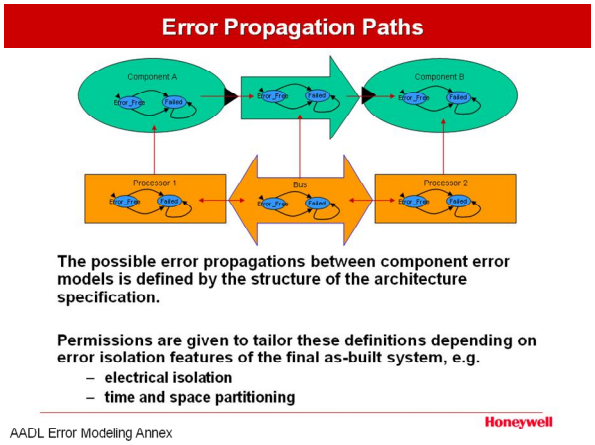


Figure 7 Architecture Integrates Error Models

Some of the supported analysis approaches include hazard analysis, failure modes and effects analysis (FMEA), fault trees, and Markov processes. Honeywell has demonstrated error modelling (hazard and FMEA) using annex capabilities on a large aircraft system [12]. ASSERT has modelled a dual redundant fault tolerant computer system using the annex [13,14].

Error models are used to develop multiple types of system dependability models and these can be analyzed within a single specification allowing consistency checking between them. See figure 8.

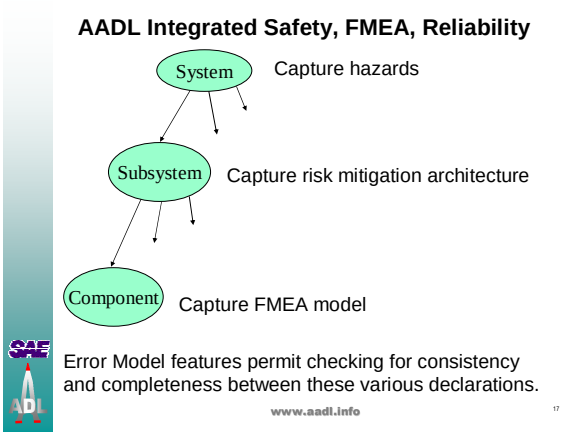


Figure 8 Integrating Dependability Analysis

Honeywell has also done analysis on multiple aircraft platforms with a variety of buses, system synchronization approaches, and sizes. These analyses included resource consumption on processors and buses, global scheduling across multiple scheduling paradigms, switched network tuning, and FMEA and Hazard analysis [12]. These analyses and others among AADL users have had a direct impact on the AADL standard, demonstrating and refining capabilities for industrial application. See figure 9.

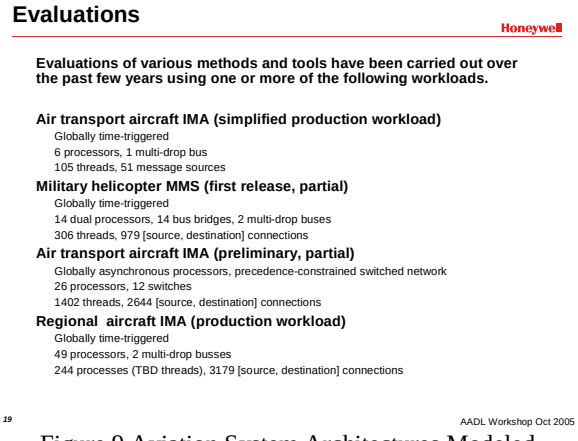


Figure 9 Aviation System Architectures Modeled

An aviation system AADL specification is available with instructions to demonstrate incremental, multi-fidelity analysis across multiple analysis dimensions. It has been posted on the SEI web site [15]. It uses the Open Source AADL Toolset Environment (OSATE) editor and analysis plug-ins. There are a number of AADL tools, many with analysis capability listed on the web site [9]. Specifications can be moved from tool to tool to leverage analysis capabilities. In some cases the tools are designed to work together for tighter integration, but well defined semantics and a significant core capability have enabled specifications to be moved from toolset to toolset to leverage analysis as demonstrated on ASSERT.

An important aspect in integrating multiple analysis approaches is the standardized AADL meta-model and XML schema. It provides a standard database format for analysis tool interaction. Tools can not only read from the XML models but also post back into it. For instance, a scheduling and binding toolset would be used to optimize processor and bus utilization given the constraints on binding captured in the AADL and the properties of threads and communication overheads. Given that binding, a reliability, latency or safety analysis could follow. Finally, from the system instance, the system could be automatically integrated with generation of glue code. See figure 10.

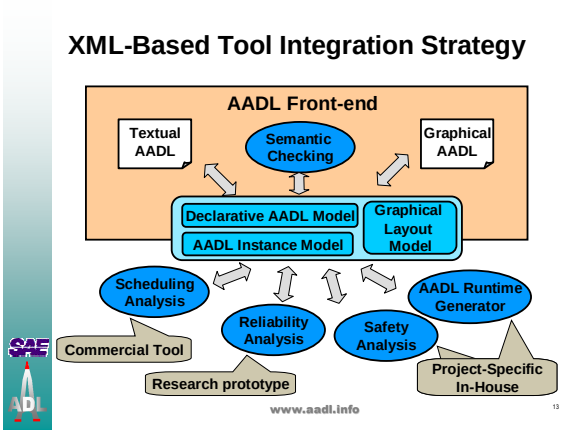


Figure 10 : Standardized and Simplified Tool Integration

## 5. Significant Industrial Initiatives and Research Drive Future Capability

The AADL standard's capability and extensibility has attracted a number of industrial initiatives and research projects on embedded systems analysis and verification. Most of these projects are cooperating with the AADL standardization committee to identify new capabilities needed in the language as well as developing analysis and code generation tools, and engineering methods. Industrial initiatives are piloting the technology to work out changes in processes to accommodate MBE using AADL, including acquisition, development, and certification. The ASSERT and the TOPCASED initiatives have already made significant contributions to the AADL language and tools. See figure 11.

The Support for Predictable Integration of mission Critical Embedded Systems (SPICES) project [16], as described in their presentation to the AADL committee, is strongly oriented to AADL modelling and analysis for use in containers which would be integrated to form systems. "The principal goals of SPICES are to extend the capacities of the microCCM component-based framework and to couple it with an AADL modeller in order to offer to system architects, software architects, and application designers a component-based modelling, design and analysis environment for distributed real-time embedded systems that should be deployed over heterogeneous targets such as GPP, DSP or FPGA." See Figure 12.

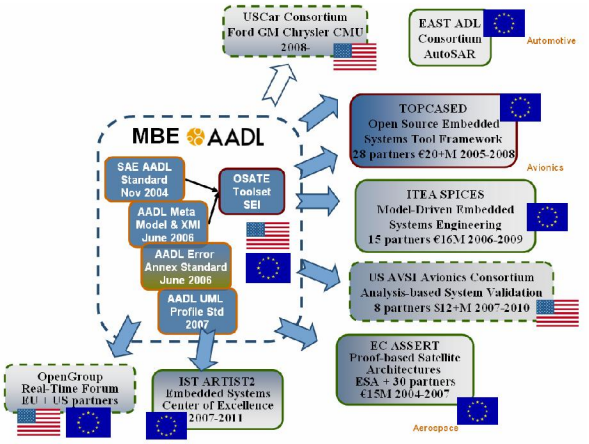


Figure 11 Industrial Initiatives Using the AADL

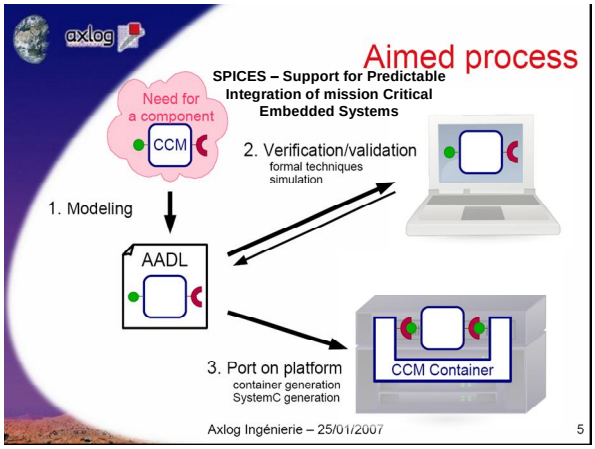


Figure 12 AADL modeling and Container Concept

Figure 13 provides an overview of three of the work packages. The first is to develop needed extensions to the AADL. These typically would be annexes like the Behaviour Annex Airbus is currently developing, but also corrections, extensions to the current standard where needed. The second work package is the development of additional AADL modelling tools and integrated verification and validation tools. The third work package is the generation of validated FPGA's and validated software component containers and how to build tools to port them onto the platform. Two work packages not shown on the figure focus on pilots and dissemination.

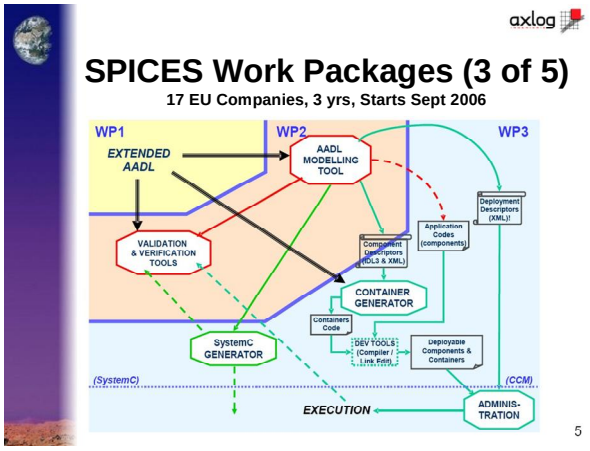


Figure 13 SPICES Extends AADL through System Building

Another new industrial initiative with a strong AADL focus is sponsored under the Aerospace Vehicle Systems Institute (AVSI) System and Software Integration Verification (SSIV) work tasks [17] and lead by Boeing. The AVSI is a cooperative of aerospace companies, DoD, and FAA to improve the integration of complex subsystems in aircraft. This AVSI project includes not only US contractors and system integrators but also European as well. Airbus and Dassault are joining with Boeing, Lockheed Martin and a number of US suppliers. There is an opportunity for European suppliers to also participate.

The theme of the AVSI-SSIV is "Integrate – then Build". It's objectives are listed in figure 14, and includes integrated analysis of the system, software, and hardware for performance, safety, security, and functionality. The prediction of system behaviour by analysis to assure acceptability occurs before the system building process begins through virtual integration. Some requirements are determined through analysis. These analyses will at least maintain existing levels of safety and security but are expected to improve them, since analysis and formal modelling will find issues testing and simulation often miss. This approach moves the industry beyond process toward evidence for safety and security.

As can be seen in figure 15, this approach modifies the acquisition process. The system integrator provides virtual models of "parts" to potential suppliers, who respond with their models of these subcomponents that fit the requirements they are bidding. Then these components are virtually re-integrated, with feedback, by the system integrator. This approach fits with investments made in Europe with the AADL and Model Based Engineering.


Figure 14 AVSI-SSIV Objectives

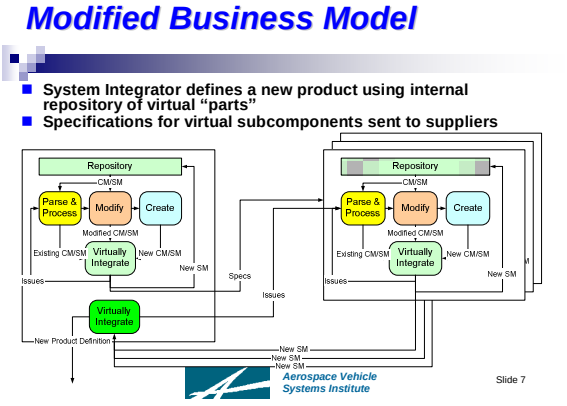


Figure 15 Virtual Integration Based Acquisition

Figure 16 illustrates the way the model bus will be used to integrate models, exchange information between tools and between the system integrator and the suppliers. The model bus plays an important role by keeping the approach from depending on specific tools. It allows the integration

of tool results into the models and the models with analysis tools. The AADL was selected after an extensive study of architecture description languages in industry and academia for its capability, existing tools, and extensibility mechanisms.

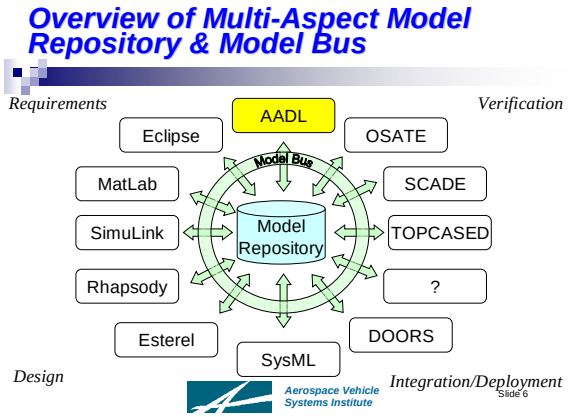


Figure 16 Integrated Tools/System Model via Model Bus & Repository

AVSI-SSIV is designed as a multi-phased program. The first phase was a planning phase and is near completion. The next phase is a one year task to demonstrate viability of the approach, which includes demonstrating AADL, analysis and the model bus and develop a new model based acquisition process for industry. This next phase will start in the spring of 2008 and is estimated at about $4M. Additional work packages are being defined for follow on research and industrial process definition, which if funded, could reach $40M.

Both the AVSI-SSIV and the SPICES programs are interested in exploring incremental certification approaches and the impact that quantitative/formal modelling can have on certification and testing.

Research programs are also a major contributor. A major advantage of the AADL is that research into new analysis methods does not depend on the development of specializations of generalized languages, such as UML, but rather the use of the standardized, well defined, architectural semantics of AADL. (UML, however, can be used to express AADL through specialization, avoiding inventing one's own custom language.) The result is that analysis tools can be leveraged across the industry and new research methods can be rapidly applied to existing AADL specifications. A number of research organizations and PhD students are leveraging the AADL for architectural analysis including safety critical avionics architecture, formal methods and assumption management at the University of Illinois under Prof. Lui Sha and Sensor Network Design and Analysis at the University of Virginia under Prof. Sang H. Son and Prof. John Stankovic. See papers at IEEE Real-Time Systems Symposium 2007. Prof. David Gluch has been applying AADL for safety, reliability and dependability analysis in real time fault tolerant systems at Embry Riddle University. The Software Engineering Institute has research projects in security analysis and fault avoidance and fault management under Dr. Jorgen Hansson and Dr. Peter Feiler. Prof. Oleg Sokolsky at the University of Pennsylvania has developed tools for AADL simulation and a process algebra based scheduling analysis toolset in association with Fremont Associates. Tools for automated integration of real-time networked systems based on AADL specifications (OCARINA) have been developed under Prof. Laurent Pautet and Prof. Jerome Hughes and advanced scheduling analysis tools (Cheddar) under Prof. Frank Singhoff at ENST ( Ecole Nationale Superieure des Telecommunications ) including integrations with TOPCASED and STOOD toolsets. Prof. Mamoun Filali of Universite Paul Sabatier, CNS, and IRIT has researched AADL and formal methods and is leading the development of the Behaviour Annex. Research at LAAS, (Laboratoire d'Analyse et d' Architecture des Systemes) contributed to the Error Modelling Annex and guidelines for its application.

The combination of research projects and industrial initiatives are bringing AADL tools and analyses to industry as well as contributing to the AADL standard.

## 6. Conclusion

The AADL's standardized component based, well defined, semantics for runtime system architecture, integrating the computer hardware and software specifically supports capturing the architecture for multi-dimension analysis. As such, it becomes an excellent place to integrate computer system engineering requirements, platform development and software engineering concerns. New research projects are likely to add new analyses and more highly automated approaches for system integration as well as new industrial acquisition processes which leverage model based engineering across multiple critical system performance dimensions.

The AADL provides or supports through tools significant model based embedded system engineering benefits. These include:

- Precise and well defined semantics supporting analyzable models to predict system performance and drive development
- Prediction of system runtime characteristics at different fidelity
- Bridge between application engineer, architect and software engineer
- Prediction early and throughout lifecycle
- Reduced integration and maintenance effort

The AADL also provides additional benefit based on its standardized features including:

- Common modelling notation across organizations
- Single architecture model augmented with properties
- Interchange & integration of architecture models
- Tool interoperability & integrated engineering environments

## 7. Acknowledgement

The author would like to thank the AADL standardization committee for their diligent efforts and willingness to share through their presentations what they are doing with the AADL. Many of the papers referenced below have been developed by committee members.

## 8. References

[1] Binns, Matt Englehart, Mike Jackson and Steve Vestal, "Domain Specific Software Architectures for Guidance, Navigation and Control," Honeywell Technology Center, Minneapolis, MN, International Journal of Software Engineering and Knowledge Engineering, Vol6, No. 2, 1996, pages 201-227.

[2] David J. McConnell, Bruce Lewis and Lisa Gray, "Reengineering a Single Threaded Embedded Missile application onto a Parallel Processing Platform using MetaH," 5t Workshop on Parallel and Distributed Real Time Systems, 1996.

[3] Patrick Farail, Pierre Dissaux, "COTRE a Software Design Workshop", DASIA 2002, May 2002.

[4] Pierre Gaufillet, « COTRE as an AADL profile », AADL Standardization Meeting, Edinburgh, July 2004

[5] Pierre Gaufillet, « TOPCASED – Toolkit In Open source for Critical Applications & SystEms Development », AADL Workshop, Paris, Oct 2005, http://www.topcased.org/

[6] Eric Conquet, « ASSERT – Automated proof-based System and Software Engineering for Real-Time systems », AADL Standardization Meeting, Edinburgh, July 2004.

[7] Society of Automotive Engineers (SAE) Avionics Systems Division (ASD) AS-2C Subcommittee. "Avionics Architecture Description Language Standard.", AS 5506, November 2004

[8] "Society of Automotive Engineers (SAE) Avionics Systems Division (ASD) AS-2C Subcommittee. "SAE Architecture Analysis & Design Language (AADL) Annex Volume 1 : Graphical AADL Notation, AADL Meta-Model and Interchange Formats, Language Compliance and Application Program Interface""; 2006 June. Report No.: AS 5506 /1.

[9] AADL Documents, OSATE Toolset, Error Modeling Plug-ins, etc, www.aadl.info

[10] Mettenburg J. "Industrial Applications of AADL - AADL Avionics Case Studies and Concepts for integrating AADL into System Development". In: AADL Standardization Meeting; 2007 January; 2007.

[11] Vestal S. "An Overview of the Architecture Analysis & Design Language (AADL) Error Model Annex ". In: AADL Workshop; 2005 October; Paris; 2005

[12] Steve Vestal, « Automating Timing and Safety Analyses from Architecture Specifications », AADL Standardization Meeting, April 2005

[13] Ana Rugina, Karama Kanoun, Mohamed Kaaniche, Jeremie Guiochet, « Dependability modelling of a fault tolerant duplex system using AADL and GSPNs », LAAS Research Report no. 05315, Draft of ASSERT Report, Sept 2005.

[14] Ana Rugina, « Dependability Modelling using AADL and the AADL Error Model Annex », AADL Workshop, Paris, Oct 2005

[15] Peter Feiler, "Evolution of an Avionics System", Paper at www.aadl.info/downloads/Models/Evolution , AADL models at www.aadl.info/downloads/Models/IntegratedModel10292007.zip

[16] Tilman J-F, Sezestre R, Schyn A. "SPICES – Support for Predictable Integration of mission Critical Embedded Systems". In: AADL Standardization Meeting; 2007 January; 2007.

[17] John Chilenski, "Aerospace Vehicle Systems Institute Systems and Software Integration Verification Overview", Nov 27, 2007, AADL Safety and Security Modeling Meeting.