Multi-Dimensional Model Based Engineering Using AADL

Bruce Lewis
US Army RD&E Command
Software Engineering Directorate
Redstone Arsenal, Huntsville Al, USA 35898
bruce.a.lewis@us.army.mil

Peter Feiler
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pa, USA 15213
phf@sei.cmu.edu

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.

1 Introduction

Given all the advances in software engineering and in engineering processes, we still have a very significant cost and risk problem to solve on large complex real time systems. We face this problem at computer system integration. There is insufficient warning of problems prior to integration. Model Based Engineering of the computer system architecture has the potential to reveal many of these integration problems earlier in the life cycle where they can be more cost effectively addressed, reducing rework and schedule delays.

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 required 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.
2 The Architecture Analysis & Design Language (AADL)

The AADL was developed for just such a purpose. It was developed from significant experimentation and research over 15 years on various DARPA programs and experiments [1,2]. 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. As shown in Figure 1, it provides the specification capability to capture the software and execution platform components, through its precise semantics provide for various system analysis methods and enable automated system integration.

The language was standardized in 2004 [3] and annexes published in 2006 for graphical notation, error modelling, standard metamodel, and Programming Language Guidelines. Version 2 of the language is now under development along with a Behavior annex based on industries desired extensions. The OMG is developing a MARTE profile for the AADL.

Because of the strong need in today’s complex systems, the capability of the AADL has stimulated large industrial initiatives to leverage the power of architectural multi-dimensional analysis and automated system integration from the specifications and models.

During the presentation, the language will be overviewed. Analysis approaches from industrial presentations will be covered to give some idea of the range of analyses that might be integrated into an architectural model. Based on an avionics architecture provided by industry, the use of the AADL in a multi-fidelity, multi dimensional development approach will be illustrated [4] from an early architectural design phase through detailed architecture description and analysis. Finally, industrial initiatives will be reviewed to show how they are building on the AADL for early system integration analysis.

3 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 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.

The fact that the AADL can be used for multiple domains of analyses has been demonstrated as illustrated in Figure 2. 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.
Sides will be presented related to some of the above analyses.

4 Industrial Initiatives

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.

The Support for Predictable Integration of mission Critical Embedded Systems (SPICES) project [5], 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.”

Another new industrial initiative with a strong AADL focus is sponsored under the Aerospace Vehicle Systems Institute (AVSI) System Architecture Virtual Integration (SAVI) work task [6]. It is 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 has joining with Boeing, Lockheed Martin and a number of US suppliers. European suppliers may join later.

The theme of the AVSI-SSIV is “Integrate – then Build”. It’s objectives includes the 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.

5 Conclusions

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. The virtual integration concept provides a new industry approach for the development of aviation systems (or automotive, health care…) and is based strongly on quantitative Model Based Engineering. The AADL supports the virtual integration concept as a backbone language for integrating multi-dimensional, multi-fidelity quantitative analysis of architectural models.

References


onto a Parallel Processing Platform using MetaH,” 5th Workshop on Parallel and Distributed Real Time Systems, 1996.


