AADL  Concepts

SAE Architecture Analysis & Design Language

Bruce Lewis
Chair, SAE AS-2C Subcommittee
Army AMCOM SED
bruce.a.lewis@us.army.mil
256-876-3224

Peter Feiler
Technical lead, editor
Software Engineering Institute
phf@sei.cmu.edu
412-268-7790

© 2005 by Carnegie Mellon University
Outline

Need for System Engineering

- What is the AADL
- Model Based Development Process
- Standardization
- Language Overview
- Early Users
- Tool Strategy
- Summary

© 2005 by Carnegie Mellon University

www.aadl.info
Computer System Engineering Issues

• Incomplete capture of specification and design
• Little insight into non-functional system properties until system integration & test. Change impacts:
  – Performance (e.g., Throughput, Quality of Service)
  – Safety - Reliability
  – Time Critical - Security
  – Schedulability - Fault Tolerance
• System Integration - high risk
• Evolvability – very expensive
• Life Cycle Support – very expensive
• Leads to rapidly outdated components

AADL addresses these concerns
Computer System Engineering

• Requires integrated hardware and software modeling
• Many dimensions to consider
• Requires an integrated process to understand effects. (AADL - Single language to capture multiple dimensions provides a built in integration)
• Software changes occur throughout the lifecycle, can’t just do it once at the beginning.
• Integration problems are the largest single technical cause of program failure.
• AADL provides a language for capturing the architecture, analyzing critical parameters, and automating the integration process.
SAE AADL Standard
An Enabler of Predictable Model-Based Embedded System Engineering

• Language for specification of task and communication architectures of Real-time, Embedded, Fault-tolerant, Secure, Safety-critical, Software-intensive systems, hardware and software
• Fields of application: Avionics, Automotive, Aerospace, Autonomous systems, …
• Based on 15 Years of DARPA funded technologies
• Standard approved & published Nov 2004
• www.aadl.info
High Level Description of the AADL

- AADL provides a formal mechanism to capture the architecture specification. Current documentation and databases do not.
  - Formal -> provides well defined abstractions for mathematical analysis
  - ADL -> mathematical analysis of architecture
  - AADL -> mathematical analysis of real-time embedded, multiprocessor, safety critical, fault tolerant systems (hardware and software).

- AADL is standard
  - Can be used with multiple projects, multiple toolsets, government and contractor using common definitions
  - Examples – POSIX, Programming languages

- AADL is precise but abstract, incremental, layer-able, system of systems, extendable
  - Analyze at the system level you need
  - Incrementally extend detail, depth, analysis models
  - Most systems require some custom analysis

- Not using it means re-inventing it project by project and inconsistent, incomplete specialized models that require expert maintenance.
Outline

- Need for System Engineering
- What is the AADL
- Model Based Development Process
  - Standardization
  - Language Overview
  - Early Users
  - Tool Strategy
  - Summary
Model Based Software and System Engineering
Architecture and Analysis Drive Real Time System Development

- AADL models software and hardware, structure and dynamics
- Repeated system analyses track development, evolution
- Auto generation, component integration to spec/analysis
- Rapid evolution through refinement of spec / components

Design feedback

Formal modeling and analysis methods and tools

Verification

Code generation

Discipline-specific design notations and editing and visualization tools

Implementation methods and tools
Typical Software Development Process

Manual, Paper Intensive, Error Prone, Resistant to Change

Requirements Analysis
Design
Implementation
Integration

High Development & Maintenance Cost
High Risk System Integration

Little Insight
Model-Based System Engineering

Predictive Analysis Early In & Throughout Life Cycle

Architecture Modeling & Analysis

Requirements Analysis

Architecture-Driven Development

System Integration

Rapid Integration
Predictable Operation
Upgradeability
Reduced Cost
Component-Based System Engineering

System Analysis
- Schedulability
- Performance
- Reliability
- Fault Tolerance
- Dynamic Configurability

System Integration
- Runtime System Generation
- Application Composition
- System Configuration

Architecture Modeling
Abstract, but Precise

Composable Components

Application Software

Execution Platform

Predictive System Engineering
Reduced Development & Operational Cost

GPS  DB  HTTPS  Ada Runtime

Devices  Memory  Bus  Processor

© 2005 by Carnegie Mellon University
Software & System Engineering

System Engineering

- Operational System
- Physical System Model
- SysML

Embedded Software System Engineering

- Embedded Software
- Computing Platform
- Physical Component
- Application Domain

SAE AADL
Outline

- Need for System Engineering
- What is the AADL
- Model Based Development Process
- Standardization
- Language Overview
- Early Users
- Tool Strategy
- Summary
From Research to Standard

Research ADLs

• MetaH
  – Real-time, modal, system family
  – Analysis & generation
  – RMA based scheduling

• Rapide, Wright, ..
  – Behavioral validation

• ADL Interchange
  – ACME

Industrial Strength

• UML 2.0, UML-RT
• HOOD/STOOD
• SDL

DARPA Funded Research since 1990

Extensible
Real-time
Dependable
AADL Standardization Committee

- Bruce Lewis (US Army AMRDEC): Chair
- Peter Feiler (SEI): technical lead, author & editor
- Steve Vestal (Honeywell): co-author
- Ed Colbert (USC): UML Profile of AADL
- Joyce Tokar (Pyrrhus Software): Ada & C Annex

Other Voting Members

- Boeing, Rockwell, Honeywell, Lockheed Martin, Raytheon, Smith Industries, General Dynamics, Airbus, Axlog, European Space Agency, TNI Europe, Dassault, EADS, High Integrity Solutions, Ford

Coordination with

- NATO Aviation, NATO Plug and Play, French Government COTRE, SAE AS-1 Weapons Plug and Play
- OMG UML RFC and MARTE for AADL UML profile
SAE AADL Standard Status

• **Sponsored by**
  – Society of Automotive Engineers (SAE)
    • Avionics Systems Division (ASD)
      – Embedded Systems (AS2)
        » Avionics Architecture Description Language Subcommittee (AS2C)
        » [bruce.a.lewis@us.army.mil](mailto:bruce.a.lewis@us.army.mil)

• **Status**
  – Requirements document SAE ARD 5296
    • Balloted and approved in 2000.
  – Standard document SAE AS 5506
    • Published Nov 10, 2004.
  – Annex document SAE AS 5506/1
    • Balloted July, 2005
    • Standardizes AADL Metamodel/XML, Graphical Symbols, Implementation Guidelines
  – Error Modeling Annex – In ballot
  – AADL UML profile – Near ballot

• **Contact**
  – [http://www.aadl.info](http://www.aadl.info)  email: [info@aadl.info](mailto:info@aadl.info)
Extensible Language Standard

• Error Model Annex as AADL extension
  – Standardized AADL extension for fault/reliability modeling
  – FMEA, MTBF analysis
  – Balloting now!
  – Prototype in MetaH, demonstrated on large aircraft model
  – OSATE plug-in in progress

• Airbus Behavior/Contracts Annex as AADL extension
  – Concurrency behavior
  – Validation of implementation
  – Behavior annex as WG standard annex
  – Component interface guarantees
  – Starting in Oct 2005
Outline

• Need for System Engineering
• What is the AADL
• Model Based Development Process
• Standardization

Language Overview
• Early Users
• Tool Strategy
• Summary
Component-Based Architecture

- Specifies a well-formed interface
- All external interaction points defined as features
- Multiple implementations per component type
- Properties to specify component characteristics
- Components organized into system hierarchy
- Component interaction declarations must follow system hierarchy
What Is Involved In Using The AADL?

- Specify software & hardware system architectures
- Specify component interfaces and implementation properties
- Analyze system timing, reliability, partition isolation
- Tool-supported system integration
- Verify source code compliance & middleware behavior

Model and analyze early and throughout product life cycle
Predictable System Integration

- Requirements, predicted, and actual properties
- Application components designed against functional and non-functional properties
- Application code separated from task dispatch & communication code
- Consistency between task & communication model and implementation through generation
- Feedback into model parameters: refinement of estimated performance values
AADL: The Language

Components with precise semantics
  – Thread, thread group, process, system, processor, device, memory, bus, data, subprogram

Completely defined interfaces & interactions
  – Data & event flow, synchronous call/return, shared access
  – End-to-End flow specifications

Real-time Task Scheduling
  – Supports different scheduling protocols incl. GRMA, EDF
  – Defines scheduling properties and execution semantics

Modal, configurable systems
  – Modes to model transition between statically known states & configurations

Component evolution & large scale development support

AADL language extensibility
Application Components

• System: hierarchical organization of components

• Process: protected virtual address space

• Thread group: organization of threads in processes

• Thread: a schedulable unit of concurrent execution

• Data: potentially sharable data

• Subprogram: Callable unit of sequential code
system GPS
features
  speed_data: in data port metric_speed
    {arch::miss_rate => 0.001 mps;};
  geo_db: requires data access real_time_geoDB;
  s_control_data: out data port state_control;
flows
  speed_control: flow path
    speed_data -> s_control_data
      { latency => 200 us;};
properties
  arch::redundancy => 2 X;
end GPS;
Graphical Model of Task Interaction

AADL connections have precise timing semantics

From Partitions

Nav signal data

To Partitions

Guidance data

20Hz

10Hz

20Hz

5Hz

2Hz

Guidance Processing

Integrated Navigation

Nav sensor data

Fuel Flow

Nav data

FP data

Aircraft Performance Calculation

Immediate & delayed data port connections preserve determinism

Nav sensor data

Nav data

FP data
Example - Thread Execution Semantics

- Nominal & recovery
- Fault handling
- Resource locking
- Mode switching
- Initialization & finalization
Faults and Modes

• AADL provides a fault handling framework with precisely defined actions
• AADL supports runtime changes to task & communication configurations
• AADL defines timing semantics for task coordination on mode switching
• AADL supports specification of mode transition actions
• System initialization & termination are explicitly modeled
Large-Scale Development

- Component type and implementation declarations in packages
  - Name scope for component types
  - Public and private package sections
  - Grouping into manageable units
  - Nested package naming
  - Qualified naming to manage name conflicts

- Supports independent development of subsystems
- Supports large-scale system of system development
System Safety Engineering

Capture the results of
  • *hazard analysis*
  • *component failure modes & effects analysis*

Specify and analyze
  • *fault trees*
  • *Markov models*
  • *partition isolation/event independence*

Integration of system safety with architectural design
  • enables cross-checking between models
  • insures safety models and design architecture are consistent
  • reduces specification and verification effort

Supported by Error Model Annex

© 2005 by Carnegie Mellon University

www.aadl.info
Outline

• Need for System Engineering
• What is the AADL
• Model Based Development Process
• Standardization
• Language Overview

Early Users
• Tool Strategy
• Summary
MetaH Case Study at AMCOM

• Missile Application reengineered
  – Missile on-board software and 6DOF environment simulation executing on dual i80960MC, Tartan Ada, VME Boards
  – Built to Generic Missile Reference Architecture
  – Specified in MetaH, 12 to 16 concurrent processes
  – MetaH reduced total re-engineering cost 40% on first project it was used on. Missile prime estimated savings at 66%.

• Missile Application ported to a new execution environment
  – multiple ports to single and dual processor implementations
  – new processors (Pentium and PowerPC), compilers, O/S
  – first time executable, flew correctly on each target environment
  – ports took a few weeks rather than 10 months.
AMCOM Effort Saved Using MetaH

Total project savings 50%, re-target savings 90%

Benefit During Application Rewrite
Benefit During Platform Retarget

Traditional Approach
Using MetaH

Man Hours

0 1000 2000 3000 4000 5000 6000 7000 8000

Review 3-DOF Translate 6-DOF RT-6DOF Transform Test 6DOF RT-Missile Build Debug Debug Re-target
AADL In Pilot Use

Architecture Specification and Automated Timing and Safety Analysis for a Large Avionics System

Steve Vastal
Larry Stockler
Dennis Foo Kune
Pam Binns
Nitin Lamba

COTRE as an AADL

Funded by the French research department (DRET, DGA) from 2002 to 2004

Goal: Real-Time architecture verification (mainly behavioral point of view)

Exploration project aiming to develop a demonstrator of AADL

Partners: AIRBUS, TNI, IRIT, LAAS, ONERA-CERT

Analyzeable and Reconfigurable AADL Specifications for IMA System Integration

David Statezni
Advanced Technology Center
Rockwell Collins, Inc.

© 2005 by Carnegie Mellon University
Outline

• Need for System Engineering
• What is the AADL
• Model Based Development Process
• Standardization
• Language Overview
• Examples from early users

Tool Strategy

• Summary
Two-Tier Tool Strategy

• Commercial Tool Support
  – UML tool environment extension based on UML profile (Artisan, Rational Rose, ILogix)
  – HOOD/STOOD: Extension to existing modeling environment with AADL export/import (TNI Europe)
  – Analysis tools interfacing via XML (Airbus, Rockwell, ASSERT)

• Open Source AADL Tool Environment (OSATE)
  – Low entry cost solution (no cost Common Public License)
  – Multi-platform support based on Eclipse
  – Vehicle for in-house prototyping of project specific architecture analysis
  – Vehicle for architecture research with access to industrial models & industry exposure to research results
XML-Based Tool Integration Strategy

AADL Front-end

Textual AADL

Semantic Checking

Graphical AADL

Declarative AADL Model

Graphical Layout Model

AADL Instance Model

Scheduling Analysis

AADL Runtime Generator

Commercial Tool

Research prototype

Reliability Analysis

Safety Analysis

Project-Specific In-House
Performance-Critical Systems Initiative
Transition Support

- Focus on model-based embedded software systems engineering: Use of AADL
- Open source AADL Tool environment (2004-2005)
  - OSATE Toolset
  - Guide to OSATE Plug-in Development
- AADL tutorial & course (2003-2005)
  - Use of AADL notation
  - AADL & Control System Applications, Large-scale Architectures, Reliable Systems
- University collaboration
  - Embry-Riddle: AADL in course, error model prototype
  - U. Illinois grad. student internship at SEI (2005)
Outline

• Need for System Engineering
• What is the AADL
• Model Based Development Process
• Standardization
• Language Overview
• Examples from early users
• Tool Strategy

Summary
AADL Use Benefits

• Model-based embedded system engineering benefits
  • Analyzable models drive development
  • Prediction of runtime characteristics at different fidelity
  • Bridge between control & software engineer
  • Prediction early and throughout lifecycle
  • Reduced integration & maintenance effort

• Benefits of AADL as SAE standard
  • Common modeling notation across organizations
  • Single architecture model augmented with properties
  • Interchange & integration of architecture models
  • Tool interoperability & integrated engineering environments