Model Based System Development Using the Architecture Analysis and Design Language

An International Industry Standard for Embedded & Real-Time Systems

USC-CSE Executive Workshop on Model-Driven Architecture
USC Campus, Los Angeles
March 16-18, 2005

Bruce Lewis – US Army AMRDEC
Peter Feiler – Software Engineering Institute
Joyce Tolkar – Pyrrhus Software
Ed Colbert – USC
Others on slides
Purpose of Architecture Analysis and Design Language (AADL)

- Enables model based architecture centric development.
- Enables standardized abstract but precise description of real time performance critical computer architectures including both hardware and software components.
- Supports System Engineering (and SoS) throughout lifecycle – incremental refinement, multi-level abstraction, integration of analysis (latency, safety, dependability, schedulability, utilization, fault tolerance etc).
- Supports auto-integrating from a model based specification into a compliant system (given compliant components).
- Objective: Significantly reduce the cost, time, and risk of development and evolution of computer based systems.
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
• Supports Computer System Model Driven Architecture based development via precise capture, analysis and Integration
• Fields of application: Avionics, Automotive, Aerospace, Autonomous systems, …
• Based on 15 Years of DARPA funded technologies
• Standard approved & published Nov 10, 2004
• www.aadl.info
SAE AS-2C AADL Subcommittee

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

Other Voting Members

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

Coordination with

• NATO Aviation, NATO Plug and Play, COTRE, ASSERT, SAE AS-1 Weapons Plug and Play, OMG UML
Model-Based System Engineering

Predictive Analysis Early In & Throughout Life Cycle

Architecture Modeling & Analysis

- Architecture-Driven Development
- Rapid Integration
- Predictable Operation
- HW SW Evolution
- Reduced Cost

Requirements Analysis

© 2004 by Carnegie Mellon University
AADL-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

Composable Components

Application Software

Execution Platform

Predictive Embedded System Engineering
Reduced Development & Operational Cost

Software System Engineer

GPS  DB  HTTPS  Ada Runtime

Devices  Memory  Bus  Processor

© 2004 by Carnegie Mellon University

www.aadl.info
Analysis Demonstrations
Honeywell, Rockwell, Airbus, SEI

- System scheduling including processors and buses across modes of operation, also utilization and sensitivity analysis
- Reliability analysis computing the probability of system failure from component failure models in redundant systems with voting using Stochastic Automata Fault models.
- Partition isolation analysis using isolation verification algorithms derived from FAA DO-178 requirements.
- Verification of the generated system executive to a MetaH (AADL) specification using linear hybrid automata.
- Safety analysis integrating Fault Tree and Hazard models with AADL architecture model for a large aircraft system.
- System timing analysis for avionics networked processors using asynchronous 653 and Time Triggered architecture variants.
- Pattern based analysis for architectural inconsistency and performance enhancement.
- Component Contract guarantees and Behavior based architectural verification using Mealy Machines and Timed Transitions Systems for semaphore usage.
- Mission/planning constraint based triggers with MetaH AADL dynamic reconfiguration of architecture.
MetaH - A Precursor to AADL

1991 DARPA DSSA program begins
1992 Partitioned PFP target (Tartan MAR/i960MC)
1994 Multi-processor target (VME i960MC)
1995 Slack stealing scheduler
1998 Portable Ada 95 and POSIX middleware configurations
1999 Hybrid automata verification of core middleware modules

Numerous evaluation and demonstration projects, e.g.
- Missile G&C reference architecture, demos, others (AMCOM SED)
- Hybrid automata formal verification (AFOSR, Honeywell)
- Missile defense (Boeing)
- Fighter guidance SW fault tolerance (DARPA, CMU, Lockheed-Martin)
- Incremental Upgrade of Legacy Systems (AFRL, Boeing, Honeywell)
- Comanche study (AMCOM, Comanche PO, Boeing, Honeywell)
- Tactical Mobile Robotics (DARPA, Honeywell, Georgia Tech)
- Advanced Intercept Technology CWE (BMDO, MaxTech)
- Adaptive Computer Systems (DARPA, Honeywell)
- Avionics System Performance Management (AFRL, Honeywell)
- Ada Software Integrated Development/Verification (AFRL, Honeywell)
- FMS reference architecture (Honeywell)
- JSF vehicle control (Honeywell)
- IFMU reengineering (Honeywell)
**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 (specification used RT messaging in MetaH)
  - Ports took 4 weeks vs. 10 months (Ada, predefined target)
AMCOM Effort Saved Using MetaH

Total project savings 50%, re-target savings 90%
Architecture Description Languages

Research ADLs
- MetaH
  - Real-time, modal, system family
  - Analysis & generation
  - RMA based scheduling
- Rapide, Wright, ..
  - Behavioral validation
- ADL Interchange
  - ACME, xADL
  - ADML (MCC/Open Group, TOGAF)

Industrial Strength
- HOOD/STOOD
- SDL
- UML 2.0, UML-RT
AADL/UML Relationship

System Engineering

AADL Core

Security

 Dependability

AADL Annexes

UML 2.0

UML-RT

Performance Timeliness

UML 1.4

Detailed design

To Be submitted to OMG for Adoption

SysML

Embedded Software System Engineering

AADL UML Profile

To Be submitted to OMG for Adoption
UML OMG COORDINATION

• AADL UML Profile will be submitted to UML under Request for Comment rules to the Real Time committee.
  – Ed Colbert of USC developing the UML profile
  – Will be standardized by SAE
  – Will be submitted to Real Time UML committee for adoption
  – Coordinated over the last year, presentations at UML meetings
  – Future coordination
AADL Two-Tier Tool Strategy

• Open Source AADL Tool Environment (OSATE)
  – Developed by SEI
  – Low entry cost solution (no cost CPL) w full AADL coverage
  – Multi-platform support based on Eclipse
  – Supports project specific architecture analysis (Video Class on dev of analysis plug-ins)
  – Usable within commercial tools.

• Commercial Tool Support
  – AADL UML profile for UML tool extension
  – Extension to user’s/vendors modeling environment with AADL export/import
  – Analysis tools interfacing via XML or XML to native filter
  – Runtime system generation tools
XML-Based Tool Integration Strategy

AADL Front-end

Textual AADL

Semantic Checking

Graphical AADL

Declarative AADL Model

AADL Instance Model

Graphical Layout Model

Scheduling Analysis

Reliability Analysis

Safety Analysis

AADL Runtime Generator

Commercial Tool

Research prototype

Project-Specific In-House
OSATE Capabilities

• OSATE Release 0.4.0 based on Eclipse Release 3
• Online AADL help
• Text to XML & XML to text
• Syntax-sensitive text editor
• Parsing & semantic checking of full AADL
• AADL property viewer
• Syntax-Sensitive Object Editor
• Model versioning & team support
• Model instantiation
• Plug-in development
  – RMA Scheduler
  – Model consistency checking
  – AADL to MetaH translator

Over 250 downloads internationally
Processed 21000 line AADL model in 10 sec

Next release
Graphical editor
Multi-file support
AADL: The Language

Components with precise definition, composition and execution semantics
- Data, subprogram, thread, thread group, process, system, processor, device, memory, bus

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

Real-time Task/System Scheduling
- Supports different scheduling protocols incl. GRMA, EDF + user defined
- Defines scheduling properties and execution semantics
- HW SW binding constraints support system optimization, families, safety

Modal, runtime reconfigurable systems
- Modes to model transition between statically known states & configurations
AADL: The Language - 2

Scaleable
- From software subprogram
- To hardware and software System of Systems

Component evolution & large scale development support
- inheritance for types and implementations
- component packages provide subcontractor support
- incremental specification
- multiple abstraction layers

AADL language extensibility
- Standard typing sublanguage for user defined types
- User/vendor/industry/standard Annex sublanguages
System Type

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
properties arch::redundancy => 2 X;
end GPS;
system implementation GPS.secure

subcomponents
  decoder: system PGP_decoder.basic;
  encoder: system PGP_encoder.basic;
  receiver: system GPS_receiver.basic;

connections
  c1: data port speed_data -> decoder.in;
  c2: data port decoder.out -> receiver.in;
  c3: data port receiver.out -> encoder.in;
  c4: data port encoder.out -> s_control_data;

flows
  speed_control: flow path speed_data -> c1 -> decoder.fs1
                 -> c2 -> receiver.fs1 -> c3 -> decoder.fs1
                 -> c4 -> s_control_data;

modes none;

properties arch::redundancy_scheme => Primary_Backup;
end GPS;
UML MDA Objectives

• Focus of UML MDA - solving the problem of rapid change in implementation approaches for standard interconnection technology and network services.

• Focus of AADL – system engineering of RT computer systems including a generative approach to the rapid change of hardware and software. Supports and goes beyond UML MDA objective.

• Supporting UML MDA in Real Time Applications – Mapping AADL to RT network service API’s for automated integration, developing properties and selecting analysis tools for performance critical aspects.
Standardized Approach Possible for AADL to UML Middleware

• Developing now an annex to standard operating systems (653 and Linux).
  – Will provide a standard concept of AADL middleware to support table driven generation to common and domain specific operating systems.

• Feasible to develop or extend annexes for UML MDA middleware (RT-Publish Subscribe, RT-CORBA …) to support generation.
Example: ARINC 653 Partitions & AADL

- ARINC Partition
- Intra-partition communication
  - Buffer
  - Blackboard
  - Semaphore
  - Event
- Inter-partition communication
  - Sampling port
  - Queuing port
- AADL process & processor abstraction
- AADL thread interaction
  - Event data port connection
  - Shared data component
  - Concurrency protocol property
  - Event port connection
- AADL process interaction
  - Data port connection
  - Event data port connection

Predeclared properties for queue processing characteristics

Project-specific properties can be added
AADL Readiness for Application

- Standard available – SAE 5506
- Language extension mechanisms standard
- Free open source full coverage smart editor, compiler
- Free open extendable toolset environment for additional tool development and integration (analysis, graphics, generation, etc)
- SEI public training - Users Guides Mid 2005
- AADL UML profile Mid 2005
- Many leading companies already using in R&D
- Commercial, private AADL tools in process
- Large avionics systems captured and analyzed
- Early Use – Std capture, prop analysis, system eng
Benefits

• Model-based system engineering benefits
  – Analyzable architecture models drive development
  – Predictable runtime characteristics at different modeling fidelity
  – Rapid model evolution & tool-based processing
  – Prediction early and throughout lifecycle
  – Reduced integration & maintenance effort

• Benefits of AADL as SAE standard
  – Common component definitions across teams, documents
  – Single architecture model augmented with analysis properties
  – Interchange & integration of architecture models
  – Tool interoperability & extensible engineering environments
  – Aligned with UML engineering, potential adoption by UML
Extra Slides for Discussion
Embedded Systems Development Concerns

• Incomplete capture of specification and design.
• Little insight into non-functional system properties until system integration & test
  – Performance (e.g., Throughput, Quality of Service)
  – Safety
  – Time Critical
  – Schedulability
  – Reliability
  – Security
  – Fault Tolerance

• System Integration – high risk
• Evolvability – very expensive
• Life Cycle Support – very expensive
• Leads to rapidly outdated components
system Data_Acquisition
features
    speed_data: in data port metric_speed;
    GPS_data:  in data port position_carthesian;
    user_input_data: in data port user_input;
    s_control_data: out data port state_control;
end Data_Acquisition;
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
Execution Platform Components

- Processor – Provides thread scheduling and execution services

- Memory – provides storage for data and source code

- Bus – provides physical connectivity between execution platform components

- Device – interface to external environment
AADL Interfaces & Connections

- **Ports**
  - Data ports
  - Event Data ports
  - Event ports

- **Connections**
  - Immediate
  - Delayed

![Diagram of AADL Interfaces and Connections](image-url)
AADL Interfaces & Connections

- **Subprograms**
  - Local
  - Server

- **Subprogram calls**
  - Local
  - Remote
Shared Data & Bus Access

- Component requires access
- Component provides access
Flow Specification in AADL

System S1

- **flow path F1**: pt1 -> pt2
- **flow path F2**: pt1 -> pt3

Flow Specification

- **flow path F1**: pt1 -> pt2
- **flow path F2**: pt1 -> pt3

Connection

System implementation S1.impl

- **flow path F5**: pt1 -> C1 -> P2.F5 -> C3 -> P1.F7 -> C5 -> pt2

Flow Implementation

- **flow path F1**: pt1 -> C1 -> P2.F5 -> C3 -> P1.F7 -> C5 -> pt2
Thread Hybrid Automata
Thread States

- **Uninitialized Thread**
- **Initialized Thread**
- **Inactive Thread**
  - Not a member of the current mode
- **Active Thread**
  - Member of the current mode

**Thread States**

- **Initialize**
- **InitializeComplete**
- **Activate**
- **ActivateComplete**
- **Deactivate**
- **DeactivateComplete**
- **Terminate**
- **TerminateComplete**
- **Finalize**
- **FinalizeComplete**
- **Dispatch**
- **DispatchComplete**
- **Compute**
- **Fault**
- **Recovered**
- **Recovered**
- **Repair**
- **Repaired**
- **Suspended**
- **Incomplete**
- **Complete**
- **NewMode**
- **InNewMode**
- **InInitMode**

**Thread State with Source Code Execution**

© 2004 by Carnegie Mellon University
Some Thread Properties

- Dispatch_Protocol => Periodic;
- Period => 100 ms;
- Compute_Deadline => value(Period);
- Compute_Execution_Time => 20 ms;
- Initialize_Entrypoint => “initialize”;
- Initialize_Deadline => 10 ms;
- Initialize_Execution_Time => 1 ms;
- Compute_Entrypoint => “speed_control”;
- Source_Text => “waypoint.java”;
- Source_Code_Size => 1.2 KB;
- Source_Data_Size => .5 KB;
Threads to Source Text and Executables

```c
main()
{
    Initialize()
    {
        //initialize variables and states
        Speed_error = 0;
        .......
    }
    static float throttle_cmd
    speed_control()
    {  
        //Computing throttle command....
        ....
        ....
        // output throttle command
        throttle_cmd = s*t;
    } ....
}
```
Thread Entrypoints & Source Text

main()
{
    Initialize()
    {
        //initialize variables and states
        Speed_error = 0;
        ........
    }
    static float throttle_cmd;
    speed_control( ) {
        //Computing throttle command....
        ....
        ....
        // output throttle command
        throttle_cmd = s*t;
    }
}
AADL Benefits

• Standard analyzable architecture description language => well defined documentation

• Supports integrated specification for application, system, and software engineering architectural requirements => avoid miscommunication

• Supports analysis of compositional effects and cross cutting impacts => manage complexity

• Supports early analysis, incremental development and model driven automated integration => reduce rework, regeneration with new HW/SW components, reduce risk and costs.
Analyzable and Reconfigurable AADL Specifications for IMA System Integration

David Statezni
Advanced Technology Center
Rockwell Collins, Inc.
Proof of Concept Example

Generic Display System with Rockwell Collin’s Switched Ethernet LAN

- Only LAN-related entities modeled
- Model generated from Input/Output & Thread data stored in Database

Model Size

- 5 Common Processing Modules
- 13 Virtual Machines
- 90 Threads
- 165 End-to-end Data Flows
CDU Subsystem Architecture

- Not modeled for this AADL example
Graphical Software (Logical) View
Architecture Specification
and
Automated Timing and Safety Analysis
for a
Large Avionics System

Steve Vestal
Larry Stickler
Dennis Foo Kune
Pam Binns
Nitin Lamba
Steve.Vestal@Honeywell.com

16 June 2004
Hardware architecture specification hand-written, based on spreadsheet and slides
   8 central processors (CPs)
   18 I/O processors (IOPs)

hw1.mh specified a switched network architecture
   12 switches
   62 point-to-point cables, mix of 10Mbs and 100Mbs
   Used for safety analysis
   Used for globally asynchronous end-to-end timing analysis

hw2.mh specified a time-triggered architecture
   8 multi-drop time-triggered busses, 10Mbs
   Used for globally time-triggered end-to-end timing analysis
   Used for globally asynchronous end-to-end timing analysis
Function/application specifications were generated from a spreadsheet listing signal data.

Total of 1322 signals to/from 40 functions (includes redundant sensors/signals).

Much data we needed was missing, we wrote the generator macro to fill this in speculatively, e.g.

- redundancy management templates
- process rates and execution times
- partition bindings

Using a dual redundancy template for all functions, total of 2644 signals to/from 80 partitions.
COTRE as an AADL profile

- Funded by the French research department (total 1.9M€, 230 m.m), from 2002 to 2004
- Goal: Real Time architecture verification (mainly from the behavioral point of view)
- Exploration project aiming to develop a demonstration tool
- Partners: AIRBUS, TNI, IRIT, LAAS, ONERA-DTIM, ENSTB
ASAAC Modelling with AADL

André Windisch
SAE AS-2 Meeting on AADL
Edinburgh, July 2004

NATO Fighter Reference Architecture
AADL and the Plug and Play Weapon
Early Experience Using the Architecture Analysis & Design Language
TC04

Yves LaCerte
3 November 2004
ASSERT

Automated proof-based System and Software Engineering for Real-Time systems

Eric Conquet
ESA/ESTEC
TEC-EME, Software Engineering and Standardization
Noordwijk, The Netherlands
• Related strategic objective: Embedded Systems
• Type of instrument: Integrated Project
• Number of partners: 29
• Project cost: 15 M€
• Amount of EC funding: 8.3 M€
  – Roughly 50% of the project cost (the rest is funded by the partners)
• Total duration of the project: 3 Years.
• Starting date: 1st September 2004.
Stood 5
and AADL

Pierre Dissaux, AADL meeting, Edinburgh, 12-15 July 2004
pierre.dissaux@tni-world.com
OSATE Plug-in Development

• Four part presentation series
  – Dec 2004 & Jan 2005
  – VTC, Webcast, telecon, video taped
  – Participants included
    • Airbus Industries, ENST, Axlog, TNI France
    • European Space Agency Netherlands
    • EADS Germany
    • US Army AMRDEC
    • Lockheed Martin, Rockwell Collins, Honeywell
    • USC, University of Pennsylvania
    • 21st Century Systems, Pyrrusoft
    • Bosch

• OSATE Plug-in Development Workshop
• OSATE Plug-in Development Guide
A Research Transition Platform

- SBIR contract requires use of AADL
  - Eglin AFB, 21\textsuperscript{st} Century Systems
  - Weapons Plug’n’Play compatibility analysis
- STTR contract uses AADL
  - U. Penn, Fremont Associates
  - Map hybrid control system language (Charon) into AADL