Avionics Architecture Description Language and UML

Avionics Architecture Description Language and the Unified Modeling Language

Ed Colbert
President, Absolute Software
colbert@abssoft.com
(760) 329-0612
Director, USC Software Engineering Certificate Program
colbert@usc.edu
(213) 821-1240

Based on presentation developed with Bruce Lewis, U.S. Army Aviation and Missile Command (AMCOM)

AADL Overview

- Society of Automotive Engineers (SAE) is developing standard Avionics Architecture Description Language
- Basic research funded by
  - U.S. Defense Advance Research Agency
- Based on
  - MetaH
    - Design by Honeywell for specification of real-time, fault-tolerant, securely partitioned, dynamically reconfigurable multi-processor system architectures
  - Unified Modeling Language (UML)
    - Object Management Group’s (OMG) standard language for object-oriented software development

Outline

- Problems Developing Embedded Real-Time Systems
- How Avionics Architecture Description Language Will Help
- Overview of AADL
- Draft Language Elements
- Extending UML
- Draft UML Metamodel & Profile for AADL
- AADL/UML Generic Missile Example
- AADL Standard Process
- Final Notes

Problems Developing Embedded Real-Time Systems

- Developing & maintaining has always been difficult
  - More capabilities required in each new system or upgrade
    - e.g. multimedia, situation awareness, mission simulation & training
  - Reliability, safety, & performance are constant concerns
    - Wrong or late answer could be deadly
  - During development
    - Difficult to integrate
    - Few means of assessing impact of decisions early
      - Often, by time developers perceive that system exceeds processor resources, it’s so late that adding or changing resources is expensive, if possible
      - Many projects cut back on capabilities so software fits hardware (despite increased costs of integration, maintenance, & upgrading)

Problems Developing Embedded Real-Time Systems (cont.)

- Embedded systems typically have very long lives & must be upgraded throughout
  - Capacity on original processors is soon exhausted as user needs increase
    - If not exhausted when fielded
  - Hardware becomes obsolete
  - Either condition can force expensive re-hosting of software to new hardware
    - Millions of dollars and years of effort can be spent

Problems Developing Embedded Real-Time Systems (cont.)

- Current development process
  - Manual, paper intensive, error prone, resistant to change

LA ACM/IEEE 2002 -- Ed Colbert
Problems Developing Embedded Real-Time Systems (cont.)

- Well-designed architecture is essential
  - Most architectural descriptions are
    - Informal documents
    - Usually centered on box-and-line diagrams, with explanatory prose
  - Visual conventions are idiosyncratic & usually project-specific
  - Results
    - Are only vaguely understood by developers
    - Cannot be analyzed for consistency or completeness
    - Are only hypothetically related to implementations
    - Cannot be supported by tools to help software architects with their tasks

Outline

- Problems Developing Embedded Real-Time Systems
- How Avionics Architecture Description Language Will Help
- Overview of AADL
- Draft Language Elements
- Extending UML
- Draft UML Metamodel & Profile for AADL
- AADL/UML Generic Missile Example
- AADL Standard Process
- Final Notes

What is an Architecture Description Language?

- Describe high-level designs
- Treats system as collection of connected components
  - Layout of components defines structure
  - Connectors define communication
  - Component interfaces are first-class citizens
  - Attributes narrowly define
    - semantics for component interactions, systemic behaviors, and emergent properties
- Does NOT describe algorithms, data structures or circuits

AADL is Domain-Specific Architecture Description Language

- Provides notations that support domain-specific architectural style or styles
  - Notations for common computation & communication paradigms
  - Architecture formally specified using notation or notations
- Models & methods to analysis
  - Estimate characteristics
  - Verify product characteristics
- Provides/supports domain-specific software patterns
- Library of configurable/generic components
  - Components that satisfy architecture guidelines for “plug-in” use
  - Components organized by some taxonomy

A New Engineering Paradigm

- Formal specification of architecture & properties
- Early detection: repeated system analyses
- Error elimination: automatic generation & integration
- Rapid evolution: refinement of models & components
- Managed change impact: Separation of concerns

Model-Based AADL Process

- Design feedback
- formal modeling and analysis methods and tools
- implementation methods and tools
- code generation
- discipline-specific design notations and editing and visualization tools
- verification

Requirements Analysis

Explicit Architecture Engineering Model

Rapid Integration Predictable System Upgradeability

LA ACM/IEEE 2002 -- Ed Colbert
Avionics Architecture Description Language and UML

Model-Based AADL Engineering

Model-Based Development Process using AADL

Outline

SAE AADL Based on MetaH

What is MetaH?

MetaH Language

---

LA ACM/IEEE 2002 -- Ed Colbert
MetaH Toolset

- Analyzes
  - Schedulability
  - Reliability
  - Safety
- Generates integrated, environment-specific code for
  - Application components
  - Executive
  - "Architectural glue"

MetaH Toolset (cont.)

MetaH Generated Partitioned Architecture

- Strong Partitioning
  - Timing Protection
  - OS Call Restrictions
  - Memory Protection
- Portability
  - Application Components
  - Tailored MetaH Executive
  - MetaH Kernel

MetaH Target Environments

- Current
  - Tartan 80960 MP
  - Standard Ada95
  - Aonix Pentium MP
  - GNAT NT
  - GNAT Solaris
  - Ada, C, C++ processes
  - GNAT RT Solaris
  - Green Hills.VxW.PowerPC
- In Process
  - LynxOS MP
    - ICC.LynxOS.PowerPC
    - Aonix.LynxOS.PowerPC
  - Green Hills.Integrity
    - 750 PowerPC
    - 750 PowerPC MP

MetaH History

- 1991 DARPA DSSA program begins
- 1992 First partitioned target operational (Tartan MAR/i960MC)
- 1994 First multi-processor target operational (VME i960MC)
- 1998 Portable Ada 95, POSIX executive configurations
- 1991—present Evaluation & demonstration projects
  - See next page

MetaH Evaluation & Demonstration Projects

- Missile G&C reference architecture (AMCOM SED)
- Missile Re-engineering demonstration (AMCOM SED)
- Space Vehicle Attitude Control System (AMCOM SED)
- Reconfigurable Flight Control (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 OWE (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)
Avionics Architecture Description Language and UML

9 January 2002

Effort Saved on AMCOM Generic Missile Project Using MetaH
- Total project 50%
- Port Phase 90%

MetaH Benefits
- Structure & behavior captured in single model
- Architecture can be optimized early & incrementally
- Highly portable software with strong isolation from hardware, operating system, & compiler dependencies
- Conformance between specified architecture model & implemented system architecture
- Significantly reduced cost for IV&V & recertification on upgrades

- Due to strong partitioning (memory and time).
- Extensions for kernel certification automation & testing support will further automate certification process
- Architecture changes made in ADL
  - Cost effective
    - e.g. very rapid processor upgrades
    - Allows low cost rapid system evolution
    - e.g. retarget SW to new bus, CPU, I/O
- Rapid retargeting allows “Software First” approach

AADL v0.1 (MetaH) Language Summary

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>Interface is a non-primitive connection interface. Implementing an interface requires a method that signals the interface is active.</td>
</tr>
<tr>
<td>Type</td>
<td>Type declares a type and may be primitive or non-primitive. The primitive type is a syntactic construct that is used as the basis for a type declaration. The non-primitive type is a user-defined type that is used to declare data objects.</td>
</tr>
<tr>
<td>Component</td>
<td>Component is a non-primitive component. Components are used to declare data objects and to define the behavior of a system. Components can be composed of subcomponents, and each component can have multiple instances.</td>
</tr>
</tbody>
</table>

Application Structure
- Application declarations combine a software architecture with a hardware architecture
- An architecture consists of communicating, typed objects
- Many attributes of the software objects can be made conditional on the type of hardware object being used

AADL Hierarchical and Compositional
- Interface to A-type objects
- There can be several implementations for the interface

Graphical MetaH Example: Top Level

LA ACM/IEEE 2002 -- Ed Colbert
Graphical MetaH Example: Application Implementation

Source Descriptions and Composition

Mapping Software Objects to Source Files

A Process Is . . .

Process Attributes

Process States
A process is implemented as
- An Ada or C procedure and,
- If process has ports, a port package.

E.g. for a process called P1 that has no ports

```c
#include <metah.h>
void p1(void)
{
    /* local declarations */
    /* initialize process */
    while (metah_true) {
        /* wait for period: */
        metah_await_dispatch();
        ...
        /* do periodic activity */
    }
}
```

Ports allow processes to exchange information via messages
- Processes, macros, modes, unshared monitors, and unshared packages, and subprograms may contain ports
- Ports are strongly typed
  - Connected ports must have same type
- Ports are either in or out

Every connection
- Effectively,
  - Starts at out port of a process
  - May actually started at port of nested subprogram
  - Ends at in port of a process
  - May actually end at port of nested subprogram
- Connections between components at same level must be from out port to in port
- Connections between container & component must be between ports with same direction
  - i.e. in to in, out to out

Imply sequencing & timing requirements
- Single sample delay
  - Transfer of data at deterministic time
    - Data is copied
      - From out port at sending component’s deadline
    - To in port at start of receiving component’s next execution
    - Allows feedback loops
- Undelayed
  - Data copied
    - From out port completion of sending component’s execution
    - To in port at start of receiving component’s next execution
  - Constrains order of execution & communication
  - Allow end-to-end deadlines, minimize latency
  - Must be acyclic, criticlity monotonic
- Connections between periodic processes have deterministic timing and function dependence

Events are sent by processes
- An event can trigger execution of one or more aperiodic processes
- Mode changes are initiated by events
A Macro Is . . .

- Hierarchical group of related processes (and other macros and modes)
- It allows connections between its own ports and events, and the ports and events of its implementation objects
- Allows equivalency between shareable objects of the contained object and of the macro itself.
- Allows setting attributes for components, connections, equivalences and the macro itself.

A Mode Is . . .

- A configuration of active processes.
- Mode changes stop and start subsets of processes and change patterns of message and event connections.
- There is always an initial mode.
- Event connections create a mode transition diagram.

Submodes are Modes Inside Modes

Possible configurations: A.X A.Y B.U B.V

Concurrent Modes . . .

Possible configurations: A.X+B.U A.X+B.V A.Y+B.U A.Y+B.V

(Not yet supported.)
Hardware Object Types

- SYSTEM: collection of processors and devices
- PROCESSOR: hosts one executable image
- DEVICE: Active device with ports and events, but cannot host an image
- CHANNEL: Hardware support for port-to-port messages, shared between processors and devices
- MEMORY: Object to which monitors, packages can be bound; shareable between processors and devices
- PORT, EVENT, MONITOR: Hardware (platform) versions of the corresponding software objects

Processor Specification

- Describes a computer
  - built around a particular type of CPU
  - running a particular μ-kernel or RTOS
  - targeted by a particular cross-development toolset
- Multiple processors may be present in a MetaH specification
- Name of a processor is used to select software implementations bound to it (i.e., ADA95)

Graphical MetaH Example: Hardware Implementation

Outline

- Problems Developing Embedded Real-Time Systems
- How Avionics Architecture Description Language Will Help
- Overview of AADL
- Draft Language Elements
- Extending UML
- Draft UML Metamodel & Profile for AADL
- AADL/UML Generic Missile Example
- AADL Standard Process
- Final Notes

Extending UML

- UML provides modeling concepts & notations for typical software modeling projects
- Users may need
  - Additional features and/or notations
  - Non-semantic information attached to models
- UML core concepts can be extended or specialized by users
  - 3 built-in extension mechanisms
    - Stereotype
    - Constraint
    - Tagged Value
  - Can be used separately or together
- Can extend UML metamodel by explicitly adding new metaclasses & other meta-constructs
  - Depends on modeling tools or use of meta-metamodel facility

UML Profile

- Predefined set of stereotypes, tagged values, constraints, & notation icons that collectively specialize UML for specific domain or process
- Does not extend UML by adding any new basic concepts
- Provides conventions for applying & specializing standard UML to particular environment or domain
- (as defined in UML 1.3)
Stereotypes

- Classify model elements at object-model level
  - Instances of stereotyped element behave as if they were instances of new metamodel classes whose form is based on existing "base" metaclasses
- Augment UML classification mechanism based on built-in UML metamodel class hierarchy
- Adds "virtual" UML metaclasses with new
  - Semantics
  - Meta-attributes
  - Property lists
  - Constraints
  - Graphical representation

Stereotypes (Cont.)

- Names of new stereotypes must not clash with
  - Names of predefined metamodel elements
  - Names of other stereotypes
- A model element can be marked by 1 stereotype
  - Also called "classified by" or "stereotyped"
  - Stereotype can be constructed as specialization of other stereotypes
  - Receives features & semantics defined for stereotype
- Intent is that tools & repositories be able to manipulate stereotyped element
  - Same as ordinary element for most editing & storage purposes
  - Differentiating it for certain semantic operations, such as well-formedness checking, code generation, or report writing

Property Lists & Tagged Values

- Any modeling element may have "arbitrary" information attached in form of property list
- Property List consists of tag-value pairs
  - Tag is user-definable unique name string for property
  - Value is string
    - "Arbitrary" from UML's perspective
    - May be constrained by definer
    - May be meaningful to tools
- Stereotype may require specific
  - Set of tags
    - "pseudo-attributes"
  - Optional default values
  - "constraints"

Constraints

- Semantic condition or restriction
  - Boolean expression associated with model element(s)
    - Must be true for the model to be well formed
    - Assertion not an executable statement
    - Certain constraints are predefined in UML
- 3 forms
  - Invariant
  - Precondition
  - Postcondition
- May be expressed in UML’s Object-Constraint Language (OCL)
- May be associated with specific stereotype to define semantics
  - Inheritable

Outline

- Problems Developing Embedded Real-Time Systems
- How Avionics Architecture Description Language Will Help
- Overview of AADL
- Draft Language Elements
- Extending UML
  - Draft UML Metamodel & Profile for AADL
  - AADL/ UML Generic Missile Example
  - AADL Standard Process
  - Final Notes

Draft Metamodel & Profile for AADL

- Following UML Class Diagrams show
  - Metamodel for MetaH concepts
  - Hierarchy of stereotypes defined for MetaH concepts
- Most MetaH concepts defined as stereotypes of UML class
  - Some MetaH concepts semantically closer to more specialized UML concepts, but
    - May be semantic incompatibilities
    - Some tools restrict
      - Types of diagrams that more specialized UML concepts can appear on
        - Kinds of relations that can be drawn between the more specialized UML concepts
- MetaH component defined as a stereotype of UML feature
  - UML feature supports current way that Graphical MetaH represents components
  - MetaH component semantically closer to a UML association
- MetaH attribute is represented as a UML property
- Need deeper study to resolve issues
Overview

Core Concepts

Primitives

Implementation Units

Relation

UML Model of AADL v0.1 (MetaH)

Core Concepts

Class

MetaH-Interface

MetaH-Type

Port

MetaH-Event

MetaH-Event-Port

MetaH-Communication-Element

Interface

MetaH-Software-Interfaces

MetaH-Hardware-Interfaces

Application-Interface

Composite Module Interface

Source Module Interface

Error-Model-Interface

Mode-Interface

Macro-Interface

Software Interfaces

Source Module Interfaces

Subprogram-Interface

Monitor-Interface

Process-Interface

MetaH-Type-Package

MetaH-Software-Interfaces

MetaH-Hardware-Interfaces

MetaH-Software-Interfaces

MetaH-Hardware-Interfaces

Application-Interface

Composite Module Interface

Source Module Interface

Error-Model-Interface

Mode-Interface

Macro-Interface

Source Module Interfaces

Subprogram-Interface

Monitor-Interface

Process-Interface

MetaH-Type-Package

MetaH-Software-Interfaces

MetaH-Hardware-Interfaces

Application-Interface

Composite Module Interface

Source Module Interface

Error-Model-Interface

Mode-Interface

Macro-Interface

Source Module Interfaces

Subprogram-Interface

Monitor-Interface

Process-Interface

MetaH-Type-Package
Problems Developing Embedded Real-Time Systems
How Avionics Architecture Description Language Will Help
Overview of AADL
Draft Language Elements
Extending UML
Draft UML Metamodel & Profile for AADL
AADL/UML Generic Missile Example
AADL Standard Process
Final Notes

UML: Top Level

UML: Application Architecture
Class Diagram

UML: Application Architecture
Collaboration Diagram

UML: Mode-Interface Class Diagram

UML: System-Interface
Class Diagram
Outline

- Problems Developing Embedded Real-Time Systems
- How Avionics Architecture Description Language Will Help
- Overview of AADL
- Draft Language Elements
- Extending UML
- Draft UML Metamodel & Profile for AADL
- AADL/UML Generic Missile Example
- AADL Standard Process
- Final Notes

AADL Standard Progress

AADL Standard Progress

Requirements Document ARD5296 approved.
Current draft of the Standard about 1/3 complete
about 100 pages.
New requirements beyond MetaH baseline being evaluated for inclusion in first or second version of standard.
AADL version 1 goal - Language Standard out by end of 2003 as predicted last year
On track.
Existing MetaH language & toolset
- Provides means for experimentation and application
- Useful for analysis and system building
- Requesting funding for prototyping of new AADL features.
Avionics Architecture Description Language and UML

AADL Standard Progress ...
- Open source graphical front end phase 1 completed with MetaH graphics
- CD Training course completed for MetaH baseline.
- Coordination with other committees: Open Group Architecture Committee, SAE GOA and Avionics OS API, POSIX, NATO ASC, Real-Time UML
- UML study being extended to provide UML Profile for AADL
  - Will be appendix in standard.

AADL Strategy

Some MetaH-Related Research Directions

Would you like to participate?
- Your participation is welcome.
  - Liaisons
  - Members need to attend 2 meetings per year to have voting privilege
- Contact: Bruce Lewis at bruce.lewis@sed.redstone.army.mil, phone 256-876-3224

Outline
- Problems Developing Embedded Real-Time Systems
- How Avionics Architecture Description Language Will Help
- Overview of AADL
- Draft Language Elements
- Extending UML
- Draft UML Metamodel & Profile for AADL
- AADL/UML Generic Missile Example
- AADL Standard Process
- Final Notes

AADL Summary
- MetaH is an Architecture Description Language and multiple tools for embedded systems, especially for Avionics systems
- It provides a means to:
  - Specify the software and hardware architecture
  - Prototype incrementally to specification
  - Analyze formally the architecture
  - Implement the final system, integrating the components and auto build the system executive and glue code
  - Evolve system rapidly
    - Within development
    - Across the lifecycle
Potential AADL applications

- Architecture specification and Component Integration toolset for product line architecture reusable components
- Adaptable workstation simulation, transitioning to tactical embedded system without loss of fidelity
- Retargeting approach for re-engineering embedded systems
- Processing environment risk reduction (SW 1st).
- Worst case schedulability analysis of complex embedded RT.
- Architecture generation to go with Component generation for very rapidly evolvable or field programmable systems.
- Building Open Architecture Avionics Systems and partitioned Flight Control, reducing Validation and Verification cost.

More Information or Help

- Information and evaluation copy application is on www.htc.honeywell.com/metah
- Two to Three day training course available through Honeywell, call Steve Vestal at 612-951-7049.
- Join SAE Aviation Architecture Description Language Task Group, call Bruce at 256-876-3224.

Next SAE AADL Meeting

- Society of Automotive Engineers Avionic Systems Division, AS-5 Embedded Computing Systems, ADL Subcommittee
- Date: 30-31 January 2002
  - Wed. and Thursday morning
- Location:
  - Raytheon
  - 2000 E. Imperial Highway
  - El Segundo, CA
- Contact:
  - Diane B Schleicher <dbschleicher@west.raytheon.com>
- Website
  - http://forums.sae.org/access/dispatch.cgi/TEAAS5_pf
  - Click Future Meetings