Extending the Open Source SAE AADL Toolset

Peter Feiler
Software Engineering Institute
phf@sei.cmu.edu
412-268-7790
Outline

- Background: SAE AADL Standard
- XML-based tool interoperability and AADL extension
- An enriched embedded system engineering environment
- Open source toolset extension: an example
- Summary
SAE Architecture Analysis & Design Language

• Notation for specification of task and communication architectures of Real-time, Embedded, Fault-tolerant, Secure, Safety-critical, Software-intensive systems
• Fields of application: Avionics, Automotive, Aerospace, Autonomous systems, …
• Based on 15 Years of DARPA funded technologies
• Standard approved by SAE in Sept 2004
• www.aadl.info
Model-Based Engineering

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

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

Model the Architecture
Abstract, but Precise

S A E  A A D L

Composable Components

Application Software

Execution Platform

GPS  DB  HTTPS  Ada Runtime

Devices  Memory  Bus  Processor

www.aadl.info
The AADL Standard

- Requirements document SAE ARD 5296
  - Input from aerospace industry
  - Balloted and approved in 2000
- SAE AADL document SAE AS 5506
  - Core language approved by SAE Sept 2004
- In review to be balloted late 2004
  - Graphical AADL notation
  - UML profile of AADL for UML1.4 and UML 2.0
  - XMI domain model, XML schema
  - Ada and C Annex
- In development
  - Error Model Annex
  - ARINC 653 Annex
Two-Tier Tool Strategy

• Open Source AADL Tool Environment (OSATE)
  – Developed by SEI
  – Low entry cost solution (no cost CPL)
  – Multi-platform based on Eclipse 3.0
  – Prototyping environment for project-specific analysis
  – Architecture research platform

• Commercial Tool Support
  – UML tool environment extension based on UML profile
  – Extension to existing modeling environment with AADL export/import
  – Analysis tools interfacing via XML or XML to native filter
  – Runtime system generation tools
An XML-Based Tool Strategy

AADL Front-end

Textual AADL

Semantic Checking

Graphical AADL

AADL XML

Scheduling Analysis

Reliability Analysis

Safety Analysis

AADL Runtime Generator

Commercial Tool

Research prototype

Project-Specific In-House
AADL Meta Model

• Defined in Eclipse Modeling Framework (EMF)
  – Collection of meta model packages with graphical views
  – Separate from, but close to UML profile of AADL

• XML as persistent storage
  – XMI specification from Ecore meta model
  – Generated XML schema

• In-core AADL model
  – Generated methods for AADL model manipulation
  – Edit history, deep copy, object editor, graphical editor
  – Methods to support
    • AADL extends hierarchy
    • feature “inheritance”
    • property value “inheritance”
AADL Meta Model Fragment
Extensible Language & Toolset

• Extensible language
  – New properties through property sets
  – Sublanguage extension
  – Project-specific & approved annex extensions
  – Examples
    • Error model for dependability modeling
    • ARINC 653 profile

• Extensible tool environment
  – Eclipse plug-in architecture
  – Integrated analysis capability
  – XML-based interface to external tools
  – Filter to external tool representations
OSATE Capabilities

- OSATE Release 0.3.0 based on Eclipse Release 3
- Online help
- Text to XML & XML to text
- Syntax-sensitive text editor
- Parsing & semantic checking of approved AADL
- AADL property viewer
- AADL to MetaH translator
- Syntax-Sensitive AADL Object Editor
- Model instantiation
- Model consistency checking

Over 150 downloads

Processed 21000 line AADL model

Next release Nov 2004
- Graphical editor
- Multi-file support
- Analysis plug-in development

© 2004 by Carnegie Mellon University
Architecture Consistency

Systemic issues discovered in architecture analysis
Issue detection codified as OSATE extensions

- Expected component connectivity
  - Required and optional port connections
- Miss rate of data stream
  - Accommodates incomplete sensor readings
  - Allows for controlled deadline misses
- State vs. state delta communication
  - Data reduction technique
  - Implies requirement for guaranteed delivery
- Data accuracy
  - Reading accuracy
  - Computational error accumulation
- Priority inversion
  - Manually assigned thread priorities
An Illustrative Example

• Pilot display response time
  – Determined by flow latency
• Analysis of high-level architecture
  – Abstract but precise
• Leverage timing characteristics of subsystems
  – Subsystems as partitions
  – Partition execution semantics
Observation: No direct connection between flight director and page content manager
Flight Director Command Flow

- Cockpit Display
- Display Manager
- Page Content Manager
- Flight Manager
- Flight Director

Request for new page
New page content
Response Time Analysis

• Worst-case scenario
  – Period delay per partition communication
    • DM sampling latency (max. = partition period)
    • Six periods of partition communication latency
    • DM execution latency (max. = partition period)
    • 0.4 seconds worst case response time
    • 0.3 seconds best case response time

• Single processor static timeline
  – One direction immediate, opposite direction phase delayed
  – Reduces partition communication latency to three periods

• Multiple processor synchronous system
  – Take into consideration bus/network latency

• Multiple processor asynchronous system
  – Asynchronous sampling with max. sampling latency = period
Response Time Plug-in

```java
public Object caseEndToEndFlow(EndToEndFlow etef) {
    int responsetime = 0;
    for(Iterator it = etef.getFlowElement().iterator(); it.hasNext()){
        FlowElement fe = (FlowElement) it.next();
        Subcomponent sub = fe.getFlowContext();
        IntegerValue ppv = getSimplePropertyValue("ARINC653","Partition_Period");
        if (ppv == null )
            errorReporter.reportError(sub,"Subcomponent is missing partition period");
        else
            responsetime += ppv.getScaledValue(PredeclaredProperties.MICROSEC);
    }
    etef.setSimplePropertyValue("Actual_Latency",responsetime);
    IntegerValue epv = etef.getSimplePropertyValue("Expected_Latency");
    if (responsetime > epv.getScaledValue(PredeclaredProperties.MICROSEC))
        errorReporter.reportWarning(efef,"Actual latency exceeds expected latency");
    return DONE;
}
```
public class CalcAllEndToEndResponseTime extends AaxlModifyAction {
   -- action to perform analysis on all end to end flows
   public void doAaxlAction(AObject obj){
      AObject as = obj.getAObjectRoot();
      AadlSwitch responseTimeSwitch = ResponseTime.createResponseTimeSwitch();
      connectionSecuritySwitch.processPreOrderAll(as);
      errorReporter.reportInfo(as,"Response Analysis done");
   }
}

public class CalcSelectedEndToEndResponseTime extends AaxlModifyAction {
   -- action to perform analysis selected end to end flow
   public void doAaxlAction(AObject obj){
      if (! obj instanceof EndToEndFlow) return;
      AadlSwitch responseTimeSwitch = ResponseTime.createResponseTimeSwitch();
      connectionSecuritySwitch.processPreOrderAll(obj);
      errorReporter.reportInfo(obj,"Response Analysis done");
   }
}
OSATE and External Tools

- Embry-Riddle Reliability Analysis
- AADL Extensions
  - Error model
  - Concurrency behavior
- Architecture Extraction
  - System Verification Manager (CMU)
  - Simulink/Matlab, Dymola models
- Architecture Import
  - SVM
- Architecture Export
  - MetaH, TTA
- MetaH Toolset (Honeywell)
  - Scheduling analysis
  - Reliability analysis
  - Isolation analysis
  - Runtime system generation
- TimeWiz Commercial Tool
  - Scheduling analysis
  - Execution trace analysis
- TimeWeaver (CMU)
  - Distributed resource allocation
  - Multi-platform runtime system generation
- Company In-house tools
- XML
- OMNET++
  - Network simulation
- Object Model Interface
  - Network model
- Model Export
  - Filters
  - Timing model
- Concurrency Analysis

© 2004 by Carnegie Mellon University
Final Observations

• Industry-standard architecture modeling notation for embedded systems
  – Abstract but precise modeling
  – Early and repeated predictable analysis

• Industry-standard XML interchange format
  – Model interchange between contractors
  – Interoperability of analysis and generation tools

• Low entry cost open source tool set
  – Prototyping environment for project-specific analysis
  – Architecture research platform
  – Capture of engineering insight