Model-based Engineering of Software-reliant Systems with AADL

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Peter H Feiler
May 2010
Outline

Challenges in Software-reliant Systems

Architecture-centric Model-based Engineering with AADL

Industrial AADL Initiatives
Embedded software systems introduce a new class of problems not addressed by traditional system modeling & analysis.
Late Discovery of System-Level SW Problems

Sources:
Model-based Engineering Pitfalls

The system

Independently developed analytical models

Multiple Analysis Truths due to Model Inconsistency

System models

Validation of implementation against system models

Disconnect Between Model & Implementation

System implementation
Mismatched Assumptions Lead to System-level Problems

Why do system level failures still occur despite best design methods & fault tolerance techniques being deployed in systems?
Outline

Challenges in Software-reliant Systems
Architecture-centric Model-based Engineering with AADL
Industrial AADL Initiatives
SAE Architecture Analysis & Design Language (AADL) for Embedded Systems

The System

Physical platform
Aircraft

Control Guidance

Embedded Application Software
Flight control & Mission

Deployed on
Utilizes

The Software

Computer System
Hardware & OS

AADL focuses on interaction between the three major elements of a software-intensive system based on architectural abstractions of each.
AADL: A Textual and Graphical Language
(www.aadl.info and wiki.sei.cmu.edu/aadl)

Well-defined semantics for software, computer hardware, physical system
  • Thread, process, data, subprogram, system, processor, memory, bus, device, virtual processor, virtual bus

Continuous control & event response processing
  • Data and event flow, synchronous call/return, shared access
  • End-to-End flow specifications

Operational modes & fault tolerant configurations
  • Modes & mode transition

Modeling of large-scale systems
  • Component variants & refinements, layered system modeling, packaging, abstract, prototype, parameterized templates, arrays of components and connection patterns

Accommodation of diverse analysis needs
  • Property definitions, sublanguage (annex) extensions
Component View

- Component types, implementations, and instances
- Model of system composition & hierarchy
- Software, execution platform, and physical components
- Well-defined component interfaces

Concurrency & Interaction View

- Time ordering of data, messages, and events
- Synchronization of logical resources
- Operational mode behavior, interaction behavior, error behavior
- Explicit interaction flow paths & protocols

Deployment View

- Execution platform as resources
- Binding of application software
- Specification & analysis of runtime properties, ...
AADL Annex Standard Extensions

Meta Model and XMI Interchange Standard Behavior Annex (ballot passed)
- Concurrency behavior
- Validation of implementation

ARINC 653 Annex (ballot passed)
- Define 653 architectural elements in AADL for analysis
- Generation of runtime & configuration file for 653-compliant O/S

Data Modeling Annex (ballot passed)
- Interface with data model in other modeling notation

Code Generation Annex (in review)
- API & code patterns for different programming languages
- Original annex in 2006

Error Model Annex (in revision)
- Error behavior as architecture model annotation
- Original annex in 2006
AADL Tooling

Eclipse-based OSATE toolset by SEI integrated with TOPCASED (www.aadl.info)
STOOD by ElliDiss (www.ellidiss.com)
Modeling creation via UML profile for AADL
  • via OMG MARTE or non-standard profile by prototyped for Rockwell Collins by Rhapsody
AVSI SAVI Phase 2 includes task to establish tool infrastructure with commercial tool vendors

For various tools integrated with AADL see https://wiki.sei.cmu.edu/aadl
Architecture-Centric Engineering Approach

Virtual Integration & Validation of Software System Architecture

SAE AADL Architecture Model

Auto-generated analytical models

Real-time Performance
  Execution time/
  Deadline
  Deadlock/starvation
  Latency

Data Quality
  Data precision/
  accuracy
  Temporal
  correctness
  Confidence

Resource Consumption
  Bandwidth
  CPU time
  Power
  consumption

Security
  Intrusion
  Integrity
  Confidentiality

Safety & Reliability
  MTBF
  FMEA
  Hazard
  analysis

Model-based Engineering of Software-reliant Systems
Feiler, May 2010
© 2010 Carnegie Mellon University
Well-defined Execution & Communication Semantics in AADL

- Thread execution
- Communication timing
- Mode transition
Taking System Modes into Account

• Scheduling analysis

Scheduling Analysis Marker (3 items)
- FlightGuidanceIMAConfiguration_FlightGuidance_a4_fullybound_Instance.CPU_1.cpu is schedulable with 55.7 % utilization
- FlightGuidanceIMAConfiguration_FlightGuidance_a4_fullybound_Instance.CPU_2.cpu is schedulable with 75.0 % utilization
- FlightGuidanceIMAConfiguration_FlightGuidance_a4_fullybound_Instance.CPU_3.cpu is schedulable with 54.0 % utilization

• Scheduling analysis of modal systems

Create a System Operation Mode
Enter a name for the new SOM. A name can consist of letters (A-Z, a-z), numbers (0-9), underscores (_), and dots (.).

Select a mode for each component:

Thread Binding Results
In system operation mode analysis_configurations_GNC_ADandC_sys_modalthreads_InstanceADC_processcontrol_configurations_GNC_ADandC_sys_modalthreads_InstanceADC_processAD_processing successfully bound to processors.

<table>
<thead>
<tr>
<th>Processor</th>
<th>Thread Bindings</th>
<th>Message Bindings</th>
<th>Network Capacities</th>
<th>AADL Prop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>% Load</td>
<td>% Available/Overload</td>
<td></td>
<td></td>
</tr>
<tr>
<td>analysis_configurations_G...</td>
<td>49%</td>
<td>51%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

gyro_based
End-to-end Latency in Control Systems

- Processing latency
- Sampling latency
- Physical signal latency

Impact of Scheduler Choice on Controller Stability
A. Cervin, Lund U., CCACSD 2006
Software-Based Latency Contributors

Execution time variation: algorithm, use of cache
Processor speed
Resource contention
Preemption
Legacy & shared variable communication
Rate group optimization
Protocol specific communication delay
Partitioned architecture
Migration of functionality
Fault tolerance strategy

AADL immediate & delayed connections assure deterministic sampling
AADL annex that supports various forms of reliability and safety analysis
Defines error model

- State transition diagram that represents normal and failed states
- Error models can be associated with hardware components, software components, connections, and “system” (composite) components

Error model consists of

- State definitions
- Propagations from and to other components
- Probability distribution and parameter definitions
- Allowed state transitions and probabilities
Leverage Connectivity in AADL Models

Fault propagation at the application logic level, at the hardware level, and between the two levels.

- Provides compositional model specification approach
- Architecture defines propagation paths for software and hardware
**AADL transformation**

- **ADAPT Tool** (Ana Rugina, LAAS-CNRS)
  - Packaged as an eclipse plug-in
  - Takes in AADL architecture and error behavior information
  - Converts to a general stochastic petri net
  - Outputs GSPN information to an XML file

- **ADAPT-MOBIUS Converter**
  - Takes in the ADAPT XML file.
  - Converts a GSPN to a Mobius Stochastic Activity Network
  - Outputs SAN information to an XML format.
Stood + Ocarina + ASN.1 tools demo

Seamless integration of SDL, SCADE, Simulink, C, Ada, ASN.1 and AADL

Over 200 papers by 50+ research groups integrating with AADL

See //wiki.sei.cmu.edu/aadl
Outline

Challenges in Software-reliant Systems
Architecture-centric Model-based Engineering with AADL
Industrial AADL Initiatives
The ASSERT process – “Applying Model-Driven Engineering Concepts to build High-Integrity systems in the IST-ASSERT process” by Jerome Hugues

The source code for each subsystem, as generated by the tools' code generators (as created in Stood and asn2aadlPlus)

The messages exchanged between subsystems

The encoders and decoders of the ASN.1 messages

The semantically equivalent types of the messages per tool
Interaction of SPICES tools

Comprehensive set of tools

Tools organised around open (and integrated) standards

AADL modelling language

Component model (CCM)

SPICES consortium

SPICES project status

AADL meeting, 20th of October 2008
Selected SEI Case Studies with AADL

AVSI SAVI Full Scale Proof Of Concept – 8 partners (2008-2011)
Army Federated to IMA Architecture Upgrade Impact Analysis (2009-2010)
Army AMRDEC AED Reliability Validation Framework (2009-10)
JPL Mission Data System Reference Architecture Validation (2007-09)
NASA/JPL Juno Feasibility Case Study (2009)
PEO AVN AB3 Model-based ATAM Architecture Analysis (2008)
Automotive Supplier Study of ETC Systems Family (2005)
Pattern-based CAAS Architecture Analysis (2003-04)
Unaffordable build-then-test approach due to exponential growth in SW size & complexity
Enable Discovery of Issues Earlier in Development through Virtual System Integration & Analysis

Standardized architecture language with strong semantics, a single source Reference Model supported by Model Repository and Model Bus concepts enable...

early validation process of system & embedded software behavior to reduce integration errors.
Earlier Problem Discovery through System Architecture Virtual Integration

Aircraft: (Tier 0)

Aircraft system: (Tier 1)
- Engine, Landing Gear, Cockpit, ...
- Weight, Electrical, Fuel, Hydraulics,...

LRU/IMA System: (Tier 2)
- Hardware platform, software partitions
- Power, MIPS, RAM capacity & budgets
- End-to-end flow latency

Subcontracted software subsystem: (Tier 3)
- Tasks, periods, execution time
- Software allocation, schedulability
- Generated executables

OEM & Subcontractor:
- Subsystem proposal validation
- Functional integration consistency
- ARINC 429 protocol mappings

Expanded scenarios:
- Refinement of mechanical systems
- Safety & reliability requirements
- Hazard & fault impact analysis
- Static analysis e.g. model checking
- End-to-end system validation

Analysis and Demonstration
- Propagate requirements and constraints
- Higher level model down to suppliers' lower level models
- Verification of lower level models satisfies higher level requirements and constraints

- Multi-tier system & software architecture
- Integrator & subcontractor virtual integration
Return on Investment Estimate

\[
\text{ROI} = \frac{NPV \ (\text{Cost avoidance with SAVI discounted at 10\%})}{NPV \ (\text{Cost to develop SAVI discounted at 10\%}) \times \text{Years}}
\]

Cost reduction ranges from $717M (7.8\%) to $2,391M (26.1\%) on a $9,176M new airplane project

For this application, every increase of 1\% in defect removal efficiency results in a conservative cost reduction of $22M

Scholarly estimates based on conservative assumptions

- Based on industry data from SAVI participants, model assumes development of a single large aircraft in the 2010-2018 timeframe
- Savings driven by reduction of rework via identification of requirements problems earlier in the development lifecycle
- Maintenance savings are not considered

ROI study is part of SEI collaboration with AVSI SAVI
Architecture-centric Multi-Notation MBE

- Embedded System Engineering
- System Engineering
- Application Software Runtime Architecture (task & communication)
- Physical System Architecture (interface with embedded SW/HW)
- Operational Environment (People, Use scenarios)
- Computer Platform Architecture (processors & networks)
- Electrical Engineering
- Control Engineering
- Application Software Components (source code)
  - Java, UML, Simulink
- Application Software Components (circuits & logic)
  - VHDL
- Physical Components (mechanical, electrical, heat)
  - Modelica
- Hardware Components (circuits & logic)
- Mechanical Engineering

SAE AADL coordinates with OMG SysML & OMG MARTE
SysML is a UML profile for system engineers
MARTE provides a UML profile for AADL
Opportunity for collaboration with INCOSE WGs
Conclusion

Impact of software runtime architecture on system is often discovered late in the development.

Problems occur due to mismatched assumptions.

Root cause areas can be addressed analytically based on architecture-centric models with well-defined semantics.

End-to-end validation results in reduced risk and increased confidence.

Industry & research community are investing in a standards-based architecture-centric MBE approach to virtual system integration.

Opportunity of collaboration with INCOSE WGs.
Peter H Feiler
phf@sei.cmu.edu
NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN “AS-IS” BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

Use of any trademarks in this presentation is not intended in any way to infringe on the rights of the trademark holder.

This Presentation may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.