Application of the Architectural Analysis and Design Language (AADL) for Quantitative System Reliability and Availability Modeling

Myron Hecht, Chris Vogl, and Alex Lam

Presented to
Aerotech 2009
Seattle, WA
November, 2009

© The Aerospace Corporation 2009
Outline

- Motivation
- Introducing AADL
- AADL Error Annex
- AADL Modeling Environment
- AADL transformation tool
- Sample Application
- Future Plans
Motivation

• Need: Support to make better decisions on system architectures
• Target systems: Space vehicle and other constrained computing environments
• Development phase: Architectural decisions made during the early design impact
• Decisions supported:
  – Extent and type of redundancy
  – Tradeoffs of reliability vs. Weight, power, and functional capability
  – Failure rate and recovery time requirements
  – Strategies for recovering from control and payload computing disruptions
  – Handling failure propagation and common mode failures
Introducing the Architecture Analysis & Design Language (AADL)

- Society of Automotive Engineers (SAE) Aerospace Standard AS5506 (2004)
  - Preceded by more than a decade of development under the DARPA Meta-H program
- Includes representations of software, computational hardware, and system components for
  - specifying and analyzing real-time embedded systems,
  - mapping of software onto computational hardware elements.
- Effective for model-based analysis and specification
  - Evolved from DARPA Meta H project
  - Highly structured, defined semantics allows for modeling and analysis
- Annex libraries define extensions to the core language concepts and syntax
  - Error Annex of particular interest
AADL/UML/SysML Relationship

AADL

Error Annex

Behavioral Annex

SysML

SysML UML Profile

AADL UML Profile

UML 2.0

MARTE Performance

Software and System Engineering
AADL Components (graphical representation)

- text and xml representations also defined
Major Features

• Provides a standardized textual and graphical notation for describing software and hardware system architectures and their functional interfaces
• Components have interactions, flows, and subcomponents
• Component interactions: Consists of directional flow through
  – *data ports for unqueued state data*
  – *event data ports for queued message data*
  – *event ports for asynchronous events*
  – *synchronous subprogram calls*
  – *explicit access to data components*
• Different system configurations and topologies can be represented using the AADL mode concept
AADL Error Annex

- AADL annex that supports reliability 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
AADL Error Model Example

**error model example**

features
ErrorFree: initial error state;
Failed: error state;
Fail: error event {Occurrence => poisson lambda};
Repair: error event {Occurrence => poisson mu};
Failvisible: in out error propagation {Occurrence => fixed p};
end example;

**error model implementation** example.general
transitions
ErrorFree-[Fail]-&gt;Failed;
Failed-[Repair]-&gt;ErrorFree;
ErrorFree-[in Failvisible]-&gt;Failed;
Failed-[out Failvisible]-&gt;Failed;
end example.general;

AADL Modeling Environment

• Eclipse Development Environment (Ganymede) and Eclipse Modeling Framework (EMF)
• Component plug-ins
  – TopCASED-A graphical editor to create AADL architecture diagrams (SEI, Aerospace modifications)
  – Error Model Editor graphical editor to create AADL error model diagrams (Aerospace)
  – OSATE-A AADL generator (SEI, Aerospace modifications)
  – ADAPT AADL to Stochastic Petri Net Generator (SEI/LAAS Toulouse)
  – ADAPT-M Stochastic Petri net to MoBIUS stochastic analysis network tool (Aerospace)
  – MoBIUS Quantitative Dependability modeling and prediction tool (University of Illinois, Champaign Urbana)
  – FMEAGEN FMEA Generator (Aerospace)
AADL transformation

OSATE-A ➔ Error Model Editor ➔ AADL Architecture Model ➔ ADAPT ➔ GPSN ➔ A-M Tool ➔ Stochastic Activity Network Models ➔ FMEA

• 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.
Open Source AADL Tool Environment (OSATE)

- Based on Eclipse Release 3
- Parsing & semantic checking of approved AADL
- AADL (Text) to AAXL (XML) and back
- Syntax-sensitive text editor
- Syntax-Sensitive AADL Object Editor
- AADL property viewer
- AADL to MetaH translator
- Online help
- Graphical layout editor
- Multi-file XML support
- First analysis plug-ins

More information: www.osate.org
TOPCASED Graphical Editor

- Open-Source “Meta-modeling” toolset
  - A model type or a language can be described using a meta model or a meta language
  - The Open Modeling Group (OMG) defined a 4-layer model
    - M3-Meta modeling language (ECORE – modification of MOF used by TOPCASED)
    - M2-Meta-model (schemas, AADL definition)
    - M1 – Model (AADL, finite state machine)
    - M0 – Real Object
  - AADL representation in ECORE
  - TOPCASED can graphically represent ECORE

More information: www.topcased.org
Error Model Editor

erro model implementation HW.impl

transitions

ErrorFree -> Activation_Fault;
Activation_Fault -> Transient_Error;
Activation_Fault -> Permanent_Error;
Transient_Error -> ErrorFree;
Permanent_Error -> Detection_Action End;
Detection_Action End -> In_Repair;
Detection_Action End -> Error_Non_Detect;
Error_Non_Detect -> In_Repair;
Transient_Error -> Transient_Error;
Error_Non_Detect -> Error_Non_Detect;
In_Repair -> In_Repair;
In_Repair -> ErrorFree;
end HW.impl;
MOBIUS Modeling Tool

• Developed by the University of Illinois at Urbana Champaign
• System-level performance and dependability modeling
• Based on stochastic analysis network representation

More information: www.mobius.uiuc.edu
More on Stochastic Activity Networks

- Can handle recoverable redundant systems with multiple states
- Adaptation of Petri Nets
- Evaluated using Mobius Software
  - Developed by University of Illinois at Urbana-Champaign
- Advantages:
  - Simulation-based solution; does not require assumption of exponential distribution
  - Allows for higher fidelity modeling of spacecraft recovery models
- Possible alternatives:
  - Stochastic Petri Nets
  - Markov models
Failure Modes and Effects Analysis Generator

<table>
<thead>
<tr>
<th>ID</th>
<th>Item</th>
<th>Initial Failure Mode</th>
<th>1st Level Effect</th>
<th>Transition</th>
<th>2nd Level Effect</th>
<th>Transition</th>
<th>3rd Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CTU Primary SU</td>
<td>Failure, case minor</td>
<td>SU, CTU Primary DownMinor</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CTU Primary SU</td>
<td>Failure, case major</td>
<td>SU, CTU Primary DownMajor</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>CTU Backup SU</td>
<td>Failure, case minor</td>
<td>SU, CTU Backup DownMinor</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>CTU Backup SU</td>
<td>Failure, case major</td>
<td>SU, CTU Backup DownMajor</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>CTU FMS</td>
<td>Switch, case failed</td>
<td>FMS, CTU SwitchFailed</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>CTU FMS</td>
<td>Switch, case failed</td>
<td>FMS, CTU SwitchFailed</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>CTU Backup PU</td>
<td>Failure</td>
<td>PU, CTU Terminated</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>ACU Primary SU</td>
<td>Failure</td>
<td>SU, ACU Primary Down</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>ACU Backup PU</td>
<td>Failure</td>
<td>SU, ACU Backup Down</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>ACU Primary PU</td>
<td>Failure</td>
<td>PU, ACU Terminated</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>ACU Backup PU</td>
<td>Failure</td>
<td>PU, ACU Terminated</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example: Low Earth Orbit Space System (LSS)

• Description of System
  – The system contains one Bus Control Unit (BCU) and one Payload Control Unit (PCU).
  – Each Control Unit contains two Control Channels comprised of one piece of Control Software (BCS, PCS) and one Control Processor (BCP, PCP).
  – The BCU is in hot backup with imperfect switching assumed.
  – The PCU is in cold backup with perfect switching assumed (thus it is modeled as having only one Control Channel).
  – The Payload relies on the Bus, thus whenever the Bus is in Standby, the Payload goes to Standby.
AADL Hardware/Software Architecture Representation
AADL Representation (using TOPCASED)
AADL Representation (using TOPCASED, continued)

Space Vehicle Diagram

SPCU Diagram
(next hierarchical level)
AADL Representation (using TOPCASED, continued)

BCU Diagram

Control Channel Diagram (both Primary and Backup)
CTU Software Unit Error Model
CTU Processor Unit Error Model

![Diagram showing the CTU Processor Unit Error Model]
CTU FMS Error Model
ACU Software Unit Error Model

Error Implementation Diagram: GeoBusErrorModel::GeoBusErrorModel::Error_Model::ACUS.impl / unnamed
ACU Processor Unit Error Model

ACU FMS Error Model
Error Model Implementation for BCU Backup Processor

error model implementation BCS.impl

transitions
Active -[Failing]-> ReportDown;
Active -[in Sleep]-> Standby;
ReportDown -[out BCSFail]-> Down;
Active -[in Terminate]-> Terminated;
Down -[Fail_case_Minor]-> DownMinor;
Down -[Fail_case_Major]-> DownMajor;
DownMinor -[MinorRepair]-> ReportStandby;
DownMajor -[MajorRepair]-> ReportStandby;
ReportStandby -[out BCSStandby]-> Standby;
Standby -[in Wake]-> Active;
Standby -[in SwitchFail ]-> Down;
Standby -[in Terminate]-> Terminated;
Active -[in CPUFail ]-> ReportTerminated;
ReportTerminated -[out BCSTerminate]-> Terminated;
end BCS.impl;
Stochastic Analysis Representation (product of ADAPT-M conversion)
LSS – Results of MoBIUS processing

<table>
<thead>
<tr>
<th>Performance Variable</th>
<th>Simulated Mean Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mission Duration</td>
<td>95038 hours</td>
</tr>
<tr>
<td>Space Vehicle Online Time</td>
<td>67291 hours</td>
</tr>
<tr>
<td>Payload Online Time</td>
<td>67291 hours</td>
</tr>
<tr>
<td>Payload Down Time</td>
<td>26026 hours</td>
</tr>
<tr>
<td>Bus Online Time</td>
<td>71069 hours</td>
</tr>
</tbody>
</table>
Conclusions

• A tool set using a common language between system engineering and dependability engineering
  – *Can enable better decision support*
  – *Enables tradeoffs and analyses during the early design phases, but can be used during other phases*

• AADL currently offers the best semantics
  – *Failure rate and failure mode definition*
  – *Inclusion of probability distributions*
  – *but progress is being made in other OMG-sponsored efforts*

• Tool set is based on public domain software
  – *Enables cooperative development*
  – *Less dependence on commercial vendor viability – challenging in a dynamic and small market place*
Acronyms

ADAPT: AADL Architectural models to stochastic Petri nets through model Transformation,
AADL: Architecture Analysis & Design Language
BCP: Bus Control Processor
BCS: Bus Control Software
BCU: Bus Control Unit
EMF: Eclipse Modeling Framework (part of Eclipse)
GEF: Graphical Editing Framework (part of Eclipse)
GMF: Graphical Modeling Framework (part of Eclipse)
GSPN: Generalized Stochastic Petri Net
LSS: Low Orbit Space System
MOBIUS: Model-Based Environment for Validation of System Reliability, Availability, Security, and Performance
OSATE: Open Source AADL Tool Environment (Software tool integrated into Eclipse)
PCP: Payload Control Processor
PCS: Payload Control Software
PCU: Payload Control Unit
SAN: Stochastic Analysis Network
TOPCASED: Toolkit In OPen source for Critical Applications & SystEms Development
References

• Society of Automotive Engineers (SAE) Aerospace Standard AS5506 (2004)