The Story of AADL

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Peter H. Feiler
Sept 29, 2014
Outline

The History of AADL
Challenges in Safety-critical Software-intensive systems
An Architecture-centric Virtual Integration Strategy with SAE AADL
Improving the Quality of Requirements
Architecture Fault Modeling and Safety
Incremental Virtual System Integration and Verification
Summary and Conclusion
In the Beginning

  - Data and event ports, Operational modes
  - Scheduling analysis and code generation

1994: Application to Missile Guidance System by Vestal and Lewis
  - Three Week Port to Dual Processor Hardware

1997: MetaH Style for ACME by Peter Feiler

1998: Error Model added to MetaH by Steve Vestal
  - Generation of fault trees and Markov models.

1999: Requirements for AADL
  - Industry input: packages, messages,
  - The best of MetaH and ACME
AADL-based Virtual System Integration
Evolution, Maturation and Transition

- Pilot Projects in US, Europe, Japan, China

- System Architecture Virtual Integration (SAVI) Software & Systems Engineering
  - JPL Mission Data System
  - Apache Model-based ATAM
  - COMPASS SE-SW Co-engineering

- SAE AADL Standard & Tool Support: Research Transition Platform
  - ESA ASSERT
  - JPL Mission Data System
  - Apache Model-based ATAM
  - COMPASS SE-SW Co-engineering

- US & European Research Initiatives
  - DARPA MetaH ACME
  - AADLV1 Timing
  - Software & System Co-engineering
  - Multi-team Safety
  - Requirements Assurance
  - DARPA META
  - DARPA HACMS Security
  - AADL Engineering Workbench

- Other Standards and Regulatory Guidance
  - OMG MARTE Embedded Systems
  - AADLV1 Timing
  - Software & System Co-engineering
  - Multi-team Safety
  - Requirements Assurance
  - DARPA META
  - DARPA HACMS Security
  - AADL Engineering Workbench

- Regulatory Guidance
  - NRC, FDA, UL

- Avionics Network Standards
  - System Safety Practice Standards

- Safety

- Other Standards and Regulatory Guidance

- AADL Timing

- Multi-team Safety

- Requirements Assurance

- DARPA META

- DARPA HACMS Security

- AADL Engineering Workbench

Sampling of International Efforts Leveraging SAE AADL

- **Compositional Timing Framework OSD** 2014
- **P Project Auto Code Gen** 2011-2014
- **OpenGroup Real-Time Forum EU + US partners 2008-current**
- **D-MILS Design of Secure Systems 2013 - $4.9M**
- **ITEA SPICES Model-Driven Embedded Systems Engineering 15 partners €16M 2006-2009**
- **IST ARTIST2 Embedded Systems Center of Excellence 2007-2012**
- **AVSI SAVI Analysis-based System Validation 12 partners $20M 2008-current**
- **ESA TASTE System & SW Validation & Generation 2007-2010**
- **DARPA META Complex System Engineering 2010-2012**
- **IST ARTIST2 Embedded Systems Center of Excellence 2007-2012**
- **Flex-eWare Auto Code Generation 2007-2010**
- **DARPA HACMS Security in CPS RC formal methods 2013-2015**
- **PROARTIS Partitioned RT systems 2010-2013 € 1.8M**
- **AADL Inspector Ellidiss 2010-current**
- **PARSEC Safety/security 2010-2013**
- **Integrated Clinical Environment Device Certification FDA KSU 2011-current**
- **RAMSES Avionics Workbench 2011-current $2M per year**
- **MASIW Avionics Workbench 2011-current $2M per year**
- **OPEES Formal analysis 2011-2014**
- **DARPA META Complex System Engineering 2010-2012**
- **DARPA HACMS Security in CPS RC formal methods 2013-2015**
- **Integrated Clinical Environment Device Certification FDA KSU 2011-current**
- **DARPA META Complex System Engineering 2010-2012**
- **DARPA HACMS Security in CPS RC formal methods 2013-2015**
- **Integrated Clinical Environment Device Certification FDA KSU 2011-current**
Outline

The History of AADL
Challenges in Safety-critical Software-intensive systems
An Architecture-centric Virtual Integration Strategy with SAE AADL
Improving the Quality of Requirements
Architecture Fault Modeling and Safety
Incremental Virtual System Integration and Verification
Summary and Conclusion
We Rely on Software for Safe Aircraft Operation

Even with the autopilot off, flight control computers still "command control surfaces to protect the aircraft from unsafe conditions such as a stall," the investigators said.

The unit continued to send false stall and speed warnings to the aircraft's primary computer and about 2 minutes after the initial fault "generated very high, random and incorrect values for the aircraft's angle of attack."

Embedded software systems introduce a new class of problems not addressed by traditional system modeling & analysis

Autopilot Off
A "preliminary analysis" of the Qantas plunge showed the error occurred in one of the jet's three air data inertial reference units, which caused the autopilot to disconnect, the ATSB said in a statement on its Web site.

The crew flew the aircraft manually to the end of the flight, except for a period of a few seconds, the bureau said.

Even with the autopilot off, flight control computers still "command control surfaces to protect the aircraft from unsafe conditions such as a stall," the investigators said.

The unit continued to send false stall and speed warnings to the aircraft's primary computer and about 2 minutes after the initial fault "generated very high, random and incorrect values for the aircraft's angle of attack."

The flight control computer then commanded a "nose-down aircraft movement, which resulted in the aircraft pitching down to a maximum of about 8.5 degrees," it said.

No Similar Event
"Airbus has advised that it is not aware of any similar event over the many years of operation of the Airbus," the bureau added, saying it will continue investigating.
Lexus GX 460 passes retest; Consumer Reports lifts “Don’t Buy” label

Consumer Reports is lifting the Don’t Buy: Safety Risk designation from the 2010 Lexus GX 460 SUV after recall work corrected the problem it displayed in one of our emergency handling tests. (See the original report and video “Don’t Buy: Safety Risk—2010 Lexus GX 460.”)

We originally experienced the problem in a test that we use to evaluate what’s called lift-off oversteer. In this test, as the vehicle is driven through a turn, the driver quickly lifts his foot off the accelerator pedal to see how the vehicle reacts. When we did this with our GX 460, its rear end slid out until the vehicle was almost sideways. Although the GX 460 has electronic stability control, which is designed to prevent a vehicle from sliding, the system wasn’t intervening quickly enough to stop the slide. We consider this a safety risk because in a real-world situation this could cause a rear tire to strike a curb or slide off of the pavement, possibly causing the vehicle to roll over. Tall vehicles with a high center of gravity, such as the GX 460, heighten our concern. We are not aware, however, of any reports of injury related to this problem.

Lexus recently duplicated the problem on its own test track and developed a software upgrade for the vehicle’s ESC system that would prevent the problem from happening. Dealers received the software fix last week and began notifying GX 460 owners to bring their vehicles in for repair.

We contacted the Lexus dealership from which we had anonymously bought the vehicle and made an appointment to have the recall work performed. The work took about an hour and a half.

Following that, we again put the SUV through our full series of emergency handling tests. This time, the ESC system intervened earlier and its rear did not slide out in the lift-off oversteer test. Instead, the vehicle understeered—or plowed—when it exceeded its limits of traction, which is a more common result and makes the vehicle more predictable and less likely to roll over. Overall, we did not experience any safety concerns with the corrected GX 460 in our handling tests.

Many appliances now rely on electronic controls and operating software. It turns out to be a problem for the Kenmore 4027 front-loader, which scored near the bottom in our February 2010 report. Our tests found that the rinse cycles on some models worked improperly, resulting in an unimpressive cleaning.

When Sears, which sells the washer, saw our February 2010 Ratings (available to subscribers), it worked with LG, which makes the washer, to figure out what was wrong. They quickly determined that a software problem was causing short or missing rinse and wash cycles, affecting wash performance. Sears and LG say they have reprogrammed the software on the models in their warehouses and on about 65 percent of the washers already sold, including the ones we had purchased.

Our retests of the reprogrammed Kenmore 4027 found that the cycles now worked properly, and the machine excelled. It now tops our Ratings (available to subscribers) of more than 50 front-loaders and we’ve made it a CR Best Buy.

If you own the washer, or a related model such as the Kenmore 4044 or Kenmore Elite 4051 or 4219, you should get a letter from Sears for a free service call. Or you can call 800-733-2299.

How do you upgrade washing machine software?
High Fault Leakage Drives Major Increase in Rework Cost

Aircraft industry has reached limits of affordability due to exponential growth in SW size and complexity.

70% Requirements & system interaction errors
80% late error discovery at high rework cost

20.5% 300-1000x
0%, 9% 80x
10%, 50.5% 20x
70%, 3.5% 1x

Major cost savings through rework avoidance by early discovery and correction. A $10k architecture phase correction saves $3M

Where faults are introduced
Where faults are found
The estimated nominal cost for fault removal

Sources:
NIST Planning report 02-3, The Economic Impacts of Inadequate

Software as % of total system cost
1997: 45% → 2010: 66% → 2024: 88%

Post-unit test software rework cost 50% of total system cost and growing

Total System Cost
Boeing 777 $12B
Boeing 787 $24B

© 2014 Carnegie Mellon University
Mismatched Assumptions in System Interactions

Why do system level failures still occur despite fault tolerance techniques being deployed in systems?  
Software system as hazard contributor

Embedded software system as major source of hazards
Model-based Engineering Pitfalls

Inconsistency between independently developed analytical models

Confidence that model reflects implementation

The system

System models

System implementation

This aircraft industry experience has led to the System Architecture Virtual Integration (SAVI) initiative
Why UML, SysML Are Not Sufficient

- System engineering
  - Focus on system architecture and operational environment
  - SysML developed to capture interactions with outside world, as a standardized UML profile
  - 4 pillars/diagrams: requirements, parameterics (added in SysML), structure, behavior
- Conceptual architecture
  - UML-based component model
  - Architecture views (DoDAF, IEEE 1471)
  - Platform Independent model (PIM)
- Embedded software system engineering
  - SAE AADL with well-defined semantics for SW, runtime, computer, physical system architectures
  - OMG Modeling and Analysis of Real Time Embedded systems (MARTE) as UML profile
  - xUML insufficient for PSM (Kennedy-Carter, NATO ALWI study)
Outline

The History of AADL
Challenges in Safety-critical Software-intensive systems
An Architecture-centric Virtual Integration Strategy with SAE AADL
Improving the Quality of Requirements
Architecture Fault Modeling and Safety
Incremental Virtual System Integration and Verification
Summary and Conclusion
AADL focuses on interaction between the three elements of a software-reliant mission and safety-critical systems.
The SAE AADL Standard Suite (AS-5506 series)

Core AADL language standard (V2.1-Sep 2012, V1-Nov 2004)

- Strongly typed language with well-defined semantics
- Textual and graphical notation
- Standardized XMI interchange format

**Standardized AADL Extensions**
- Error Model language for safety, reliability, security analysis
- ARINC653 extension for partitioned architectures
- Behavior Specification Language for modes and interaction behavior
- Data Modeling extension for interfacing with data models (UML, ASN.1, …)

**AADL Annex Extensions in Progress**
- Requirements Definition and Assurance Annex
- Synchronous System Specification Annex
- Hybrid System Specification Annex
- System Constraint Specification Annex
- Network Specification Annex
System Level Fault Root Causes

Violation of data stream assumptions
• Stream miss rates, Mismatched data representation, Latency jitter & age

Partitions as Isolation Regions
• Space, time, and bandwidth partitioning
• Isolation not guaranteed due to undocumented resource sharing
• fault containment, security levels, safety levels, distribution

Virtualization of time & resources
• Logical vs. physical redundancy
• Time stamping of data & asynchronous systems

Inconsistent System States & Interactions
• Modal systems with modal components
• Concurrency & redundancy management
• Application level interaction protocols

Performance impedance mismatches
• Processor, memory & network resources
• Compositional & replacement performance mismatches
• Unmanaged computer system resources

End-to-end latency analysis
Port connection consistency

Process and virtual processor to model partitioned architectures

Virtual processors & buses
Multiple time domains

Operational and failure modes
Interaction behavior specification
Dynamic reconfiguration
Fault detection, isolation, recovery

Resource allocation & deployment configurations
Resource budget analysis & scheduling analysis

Codified in Virtual Upgrade Validation method
Architecture-Centric Quality Attribute Analytics

Single Annotated Architecture Model Addresses Impact Across Operational Quality Attributes

Safety & Reliability
- MTBF
- FMEA
- Hazard analysis

Security
- Intrusion
- Integrity
- Confidentiality

Data Quality
- Data precision/accuracy
- Temporal correctness
- Confidence

Resource Consumption
- Bandwidth
- CPU time
- Power consumption

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

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

Single Annotated Architecture Model Addresses Impact Across Operational Quality Attributes

Architecture Model

Auto-generated analytical models
Multi-Fidelity End-to-end Latency in Control Systems

Common latency data from system engineering:
- 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
Early Discovery and Incremental V&V through Virtual Integration (SAVI)

**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

**System & SW Engineering:**
- Mechatronics: Actuator & Wings
- Safety Analysis (FHA, FMEA)
- Reliability Analysis (MTTF)

**OEM & Subcontractor:**
- Subsystem proposal validation
- Functional integration consistency
- Data bus protocol mappings

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

**Repeated Virtual Integration Analyses:**
- Power/weight
- MIPS/RAM, Scheduling
- End-to-end latency
- Network bandwidth

**Proof of Concept Demonstration and Transition by Aerospace industry initiative**
- 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 (in AADL)
- Incremental end-to-end validation of system properties
Outline

The History of AADL
Challenges in Safety-critical Software-intensive systems
An Architecture-centric Virtual Integration Strategy with SAE AADL
Improving the Quality of Requirements
Architecture Fault Modeling and Safety
Incremental Virtual System Integration and Verification
Summary and Conclusion
There is more to requirements quality than traceability

*IEEE 830-1998 Recommended Practice for SW Requirements Specification*
Mixture of Requirements & Architecture Design Constraints

Requirements for a Patient Therapy System

1. The patient shall never be infused with a single air bubble more than 5ml volume.
2. When a single air bubble more than 5ml volume is detected, the system shall stop infusion within 0.2 seconds.
3. When piston stop is received, the system shall stop piston movement within 0.01 seconds.
4. The system shall always stop the piston at the bottom or top of the chamber.

Typical requirement documents span multiple levels of a system architecture

We have made architecture design decisions.

We have effectively specified a partial architecture

Adapted from M. Whalen presentation
**System Specification and Requirements Coverage**

- **Developmental Requirements**
  - Modifiability
  - Assurability

- **Quality attribute utility tree**
  - Performance
  - Modifiability
  - Availability
  - Security

- **Mission Requirements**
  - Function
  - Behavior
  - Performance

- **Dependability Requirements**
  - Reliability
  - Safety
  - Security

- **Environmental Assumptions**
  - Precondition
  - Postcondition
  - Invariant

- **Interaction contract:**
  - match input assumption with guarantee

- **Implementation constraints**

- **The Story of AADL**
  - Feiler, Sept 29, 2014
  - © 2014 Carnegie Mellon University
Architecture-led Requirement & Hazard Specification

Error Propagation Ontology

Leveson pattern
Outline

The History of AADL
Challenges in Safety-critical Software-intensive systems
An Architecture-centric Virtual Integration Strategy with SAE AADL
Improving the Quality of Requirements
Architecture Fault Modeling and Safety
Incremental Virtual System Integration and Verification
Summary and Conclusion
AADL Error Model Scope and Purpose

System safety process uses many individual methods and analyses, e.g.

- hazard analysis
- failure modes and effects analysis
- fault trees
- Markov processes

Goal: a general facility for modeling fault/error/failure behaviors that can be used for several modeling and analysis activities.

Annoted architecture model permits checking for consistency and completeness between these various declarations.

Related analyses are also useful for other purposes, e.g.

- maintainability
- availability
- Integrity
- Security

SAE ARP 4761 Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment

Demonstrated in SAVI Wheel Braking System Example

Error Model Annex can be adapted to other ADLs
Error Model V2: Abstraction and Refinement

Four levels of abstraction:

- Focus on fault interaction with other components
  - Probabilistic error sources, sinks, paths and transformations
  - Fault propagation and Transformation Calculus (FPTC) from York U.
- Focus on fault behavior of components
  - Probabilistic typed error events, error states, propagations
  - Voting logic, error detection, recovery, repair
- Focus on fault behavior in terms of subcomponent fault behaviors
  - Composite error behavior state logic maps states of parts into (abstracted) states of composite
- Types of malfunctions and propagations
  - Common fault ontology
Error Propagation Contracts

Incoming/Assumed
- Error Propagation: Propagated errors
- Error Containment: Errors not propagated

Outgoing/Contract
- Error Propagation
- Error Containment

Bound resources
- Error Propagation
- Error Containment
- Propagation to resource

Legend
- Propagation of Error Types
- Direction
- Processor
- HW Binding
- Not propagated

Error Flow through component
Path: P1.NoData -> P2.NoData
Source: P2.BadData
Path: processor.NoResource -> P2.NoData

“Not“ on propagated indicates that this error type is intended to be contained. This allows us to determine whether propagation specification is complete.
System engineering activity with focus on failing components.
Discovery of Unexpected PSSA Hazard through Repeated Virtual Integration

**EGI**
- **EGI Logic**
  - **Oper’l**
  - **Failed**
  - **Corrupted**
- **EGI HW**
  - **Oper’l**
  - **Failed**

**Anticipated: No EGI data**

**CorruptedData**

**Unanticipated propagation of corrupted Airspeed data results in Stall due to miss-correction**

**Vibration causes boards to touch which causes EGI data corruption**

**EGI maintainer adds corrupted data hazard to model.**
**Error Model analysis of integrated model detects unhandled propagation.**

**Flight Mgmt System**
- **Auto Pilot**
  - **Operational**
  - **Failed**
- **FMS Processor**
  - **Operational**
  - **Failed**
- **FMS Power**

**NoData**
- **Airspeed Data**

**Anticipated: No Service**
- **Actuator Cmd**

**NoService**
- **Stall**

The Story of AADL
Feller, Sept 29, 2014
© 2014 Carnegie Mellon University
Recent Automated FMEA Experience

Failure Modes and Effects Analyses are rigorous and comprehensive reliability and safety design evaluations

- Required by industry standards and Government policies
- When performed manually are usually done once due to cost and schedule
- If automated allows for
  - multiple iterations from conceptual to detailed design
  - Tradeoff studies and evaluation of alternatives
  - Early identification of potential problems

Largest analysis of satellite to date consists of 26,000 failure modes

- Includes detailed model of satellite bus
- 20 states perform failure mode
- Longest failure mode sequences have 25 transitions (i.e., 25 effects)
Understanding the Cause and Effects of Faults

Through model-based analysis identify architecture induced unhandled, testable, and untestable faults and understand root causes, contributing factors, impact, and potential mitigation options.

Fault Impact & FDIR Analysis

Architecture Fault Model Analysis

Discover testable and untestable faults

Discover unhandled faults & safety violations

FADEC Operational Mode & Fault Mgmt Behavior Analysis

Model validation

Requirements

Faults that can be tested
Decision coverage

Faults that cannot be tested
Race conditions

Improved documentation & design

Faults that are unhandled
Transient data loss in protocol

Root Cause of Data Loss Is Non-deterministic Temporal Buffer Read/Write Ordering

Fault Impact Analysis

Detection of Unhandled Data Loss Fault

Fault propagation Effects Engine Control Mode to Issue Shut Down Engine Sequence

Reachability Analysis Of Unsafe States

Read/write Timeline Analysis Under Cyclic Executive & Preemptive Scheduler
The Symptom: Missed Stepper Motor Steps

Stepper motor (SM) controls a valve
  • Commanded to achieve a specified valve position
    – Fixed position range mapped into units of SM steps
  • New target positions can arrive at any time
    – SM immediately responds to the new desired position

Safety hazard due to software design
  • Execution time variation results in missed steps
  • Leads to misaligned stepper motor position and control system states
  • Sensor feedback not granular enough to detect individual step misses

Software modeled and verified in SCADE
  Full reliance on SCADE of SM & all functionality
  Problems with missing steps not detected

Software tests did not discover the issue
  Time sensitive systems are hard to test for.

Two Customer Proposed Solutions
  Sending of data at 12ms offset from dispatch
  Buffering of command by SM interface
  No analytical confidence that the problem will be addressed

Other Challenge Problems
  Aircraft wheel braking system
  Engine control power up
  Situational Awareness & health monitoring
Support of SAE ARP4761 System Safety Assessment Practice

- FHA Spreadsheet
  - Uses error sources

- FMEA Spreadsheet
  - Uses error flows & propagations

- AADL & EMV

- FTA
  - CAFTA, OpenFTA
  - Uses composite error behavior

- Markov Chain PRISM
  - Uses error flows & behavior

- RBD/DD
  - OSATE plugin
  - Uses composite error behavior

Component | Error | Hazard Description | Crossrefer | Functional Failure | Operational Failure
--- | --- | --- | --- | --- | ---
StabilatorPositionSel | ServiceOmission or No stabilator position readings due to input | Loss of sensor readings | 1.1.3 | All | All
StabAct1 | ServiceOmission or Failure to move stabilator into desired position | Loss of actuator functionality | 1.1.2 | All | All
StabAct2 | ServiceOmission or Success to move stabilator into desired position | Loss of actuator functionality | 1.1.2 | All | All
StabilatorControl | null on ActCond | Failure of computed data should sign | 1.1.1 | Loss of guidance values | Approach
StabilatorControl | null on ActCond | Failure of computed data should sign | 1.1.1 | Loss of guidance values | Approach
StabilatorControl | null on ActCond | Failure of computed data should sign | 1.1.1 | Loss of guidance values | Approach

Supplier Subsystem

Software Engineering Institute | Carnegie Mellon

The Story of AADL
Feller, Sept 29, 2014
© 2014 Carnegie Mellon University
Outline

The History of AADL
Challenges in Safety-critical Software-intensive systems
An Architecture-centric Virtual Integration Strategy with SAE AADL
Improving the Quality of Requirements
Architecture Fault Modeling and Safety
Incremental Virtual System Integration and Verification
Summary and Conclusion
Certification & Recertification Challenges

Certification: assure the quality of the delivered system

- **Sufficient evidence** that a **system implementation** meets **system requirements**
- **Quality of requirements** and **quality of evidence** determines quality of system

Certification related rework cost

- Currently 49% of total system cost and growing

Recertification Challenge

- Desired cost of recertification in proportion to change
Reliability & Qualification Improvement Strategy

|-------------------------------------------|-------------------------------------------------|---------------------------------------------|----------------------------------------------------------|

**Four pillars for Improving Quality of Critical Software-reliant Systems**

- **Mission Requirements**
  - Function
  - Behavior
  - Performance

- **Survivability Requirements**
  - Reliability
  - Safety
  - Security

- **Model Repository**
  - Architecture Model
  - Component Models
  - System Implementation
  - System configuration

- **Operational & failure modes**
- **Resource, Timing & Performance Analysis**
- **Reliability, Safety, Security Analysis**

2010 SEI Study for AMRDEC Aviation Engineering Directorate
Building the Assurance Case throughout the Life Cycle

Continuous Confidence Measure throughout Life Cycle that a System Meets its Requirements

Architecture-centric Virtual Integration

Architecture-Led Requirements Specification

Virtual Architecture Integration & Analysis

Code Coverage Testing

Design Validation by Virtual Integration

Flight Test

System Integration Lab Testing

Early Discovery through Architecture Analysis leads to Assurance Related Rework Reduction

Incremental Evolution and Execution of Assurance Plans

Incremental Architecture & Requirement Evolution

Auto-generation from verified models
AADL&SCADE/Simulink
Ada SPARK/Ravenscar
MISRA C

Build the System

Build the Assurance Case
Outline

The History of AADL
Challenges in Safety-critical Software-intensive systems
An Architecture-centric Virtual Integration Strategy with SAE AADL
Improving the Quality of Requirements
Architecture Fault Modeling and Safety
Incremental Virtual System Integration and Verification
Summary and Conclusion
15 Years of the SAE AS-2C AADL Committee
10 Years since the first publication of the SAE AADL standard
And many more 😊
References

AADL Website [www.aadl.info](http://www.aadl.info) and AADL Wiki [www.aadl.info/wiki](http://www.aadl.info/wiki)

Blog entries and podcasts on AADL at [www.sei.cmu.edu](http://www.sei.cmu.edu)

AADL Book in SEI Series of Addison-Wesley

On AADL and Model-based Engineering
[http://www.sei.cmu.edu/library/assets/ResearchandTechnology_AADLandMBE.pdf](http://www.sei.cmu.edu/library/assets/ResearchandTechnology_AADLandMBE.pdf)

On an architecture-centric virtual integration practice and SAVI

On an a four pillar improvement strategy for software system verification and qualification
