Incremental Verification and Validation of System Architecture for Software Reliant Systems Using AADL (Architecture Analysis & Design Language)

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Bruce Lewis (US Army), Peter H Feiler (SEI)

Layered Assurance Workshop, Dec 6, 2010
Gravity of the Issue Today

- **System Complexity is Growing Rapidly...**

**Estimated Onboard SLOC Growth**

- **Slope:** 0.1778  **Intercept:** -338.5
- **Curve Implies SLOC doubles about every 4 years**

This line fit is pegged at 27.5 M SLOC because the SLOC sizes for 2010 - 2020 are not affordable. The COCOMO II estimated costs to develop that much software is in excess of $10B.

Acronyms:

- **SLOC:** source lines of code
- **COCOMO II:** COnstructive COst MOdel II

Airbus data source: J. P. Potocki De Monteil, “Computer Software in Civil Aircraft,” Sixth Annual Conference on Software Assurance (Compass ’91), Gaithersburg, MD, June 24-27, 1991

Boeing data source: J. J. Chilenski, 2009
Late Discovery of System-Level Problems

80% of accidents due to operator error
High recertification cost of design error corrections leads to 75% of operator time spent in work-arounds

Requirements & system interaction errors

80% late error discovery at high rework & recertification cost

60% of errors in fault management software

System-level fault propagation due to incomplete/inconsistent requirements and mismatched assumptions.

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

Sources:
New Levels of System Interaction Complexity & Mismatched Assumptions – AADL addresses

System Engineer

Physical Plant Characteristics
- Lag, proximity
- Model recalibration

Control Engineer

Data Representation
- Ariane 4/5: 16-bit data
- Air Canada: gal vs. l

Application Developer

Concurrency
- Race conditions crash applications designed for single-core on multi-cores

System User

Human

Operator Error
- Driver lockout

System Under Control

Data Streams
- Unstable control & inconsistent state due to jitter and loss

Control System

Data Representation
- Ariane 4/5: 16-bit data
- Air Canada: gal vs. l

Application Software

Concurrency
- Race conditions crash applications designed for single-core on multi-cores

Compute Platform

Distribution & Redundancy
- Loss of redundancy & other hazards due to HW Virtualization

Runtime Architecture

Concurrency
- Race conditions crash applications designed for single-core on multi-cores

Software runtime system impacts safety-critical software & system properties
Fault Root Causes Due to Runtime System Architecture

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

AADL concepts capture key architecture abstractions to address root causes

Data (stream) consistency
End-to-end latency analysis

Modeling of partitioned architectures

Fault propagation
security analysis
redundancy patterns

Validation by model checking & proofs
AADL focuses on interaction between the three major elements of a software-intensive system based on architectural abstractions of each.
AADL: The Language

Designed for standardized incremental, composable, quantitative analysis and generative system integration

Precise semantics for components & interactions
  • Thread, process, data, subprogram, system, processor, memory, bus, device, virtual processor, virtual bus, abstract
  • Typed properties, properties with units and model reference values

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

Operational modes & fault tolerant configurations
  • Modes & mode transition, mode specific properties & configurations

Modeling of large-scale systems
  • Component variants, packaging of AADL models, public/private

Accommodation of diverse analysis needs
  • Extension mechanism (property set, sublanguage) standardized
AADL Annex Standard Extensions

Behavior Annex (ballot passed 2010)
  • Concurrency behavior
  • Validation of implementation

ARNIC 653 Annex (ballot passed 2010)
  • Define 653 architectural elements in AADL for analysis
  • Generation of runtime & configuration file for 653-compliant O/S

Data Modeling Annex (ballot passed 2010)
  • 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
Nominal & recovery
Fault handling
Resource locking
Mode switching
Initialization
Finalization
---------
Temporal Logic
Modes
Behavior Annex

Thread Example Diagram
Potential Model-based Engineering Pitfalls

The Issues

Inconsistency between independently developed analytical models

Confidence that model reflects implementation

Potential Solution

Architecture-centric model repository

Generation from validated models

System implementation

Security model

Fault model

Inconsistency between independently developed analytical models

System implementation

Confidence that model reflects implementation
Single-Truth through consistency across Architectural models ➞ Architecture-Centric

**SECURITY**
- Intrusion
- Integrity
- Confidentiality

**RESOURCES**
- Bandwidth
- CPU Time
- Power Consumption

**ARCHITECTURAL MODEL**

**REAL-TIME PERFORMANCE**
- Deadlock/Starvation
- Latency
- Execution Time/Deadline

**CONFIDENCE**

**Increased confidentiality requirement**
- change of encryption policy

**Key exchange frequency changes**
- Message size increases
  - increases bandwidth utilization
  - increases power consumption

**Increased computational complexity**
- increases WCET
- increases CPU utilization
- increases power consumption
- may increase latency
Formal Methods & AADL (A bridge to formal modeling from an architecture specification).

Concurrency & mode logic: interface with Alloy (deNiz)
Simulink & AADL integration: Emmeskey & Telecom Paristech
Model checking based on Simulink specifications: Rockwell Collins
Behavioral component interaction – AADL & BIP: Verimag
Formal proofs & AADL – BLESS (pace maker): Larson
AADL & Maude Model Checking: Meseguer (UIUC), U Leicester
AADL & Timed Abstract State Machines (TASM): Lundquist
AADL & Timed Automata (Cheddar): Singhoff
AADL & Process Algebra: Sokolsky
AADL & UPPAAL: Sokolsky, Lundquist
AADL & timed Petri nets: Filali (TINA), Kordon
Consistency Across Virtual Integration Models, (Nam, Sha, deNiz)
Summary – AADL Strong Semantics

Integration of CPS effects into the architecture context
Understanding of runtime behavior and communication impact
Single truth modeling and transformation
Model compilation and Model composition
Incremental Verification and Validation
Precise, correct by construction code generation
Analysis tools per domain built to common architectural semantics
Bridge to formal analysis

Cyber-Physical adds several more dimensions of complexity – to integrate the effects into an understanding of system behavior we need a similar standardized approach, perhaps AADL annex.
The ASSERT process – “Applying Model-Driven Engineering Concepts to build High-Integrity systems in the IST-ASSERT process” by Jerome Hugues

ASN.1 Code generators → System Model

ASN1 compiler (asn1c)

C code:
- Behavior
- Data structures

AADL Container

C code mapping

C code:
- Data structures
- Encoder/decoder

AADL Container

AADL Runtime
Aerospace Vehicle Systems Institute

AVSI is a global cooperative of aerospace companies, government organizations, and academic institutions.

Past AVSI projects have covered the breadth of aerospace systems and current research includes projects in the areas of reliability, certification, and virtual integration.

The System Architecture Virtual Integration program is an AVSI program addressing virtual integration of systems.

SEI was selected as the contractor to help work the proof-of-concept Effort.

June 09
How Are We to Address This Issue?

Analysis & Validation through Virtual Integration!

But what exactly does that mean?
SAVI Approach: Integrate, Then Build

**SAVI is**

- A changed acquisition paradigm to facilitate systems integration
- A research effort to define the standards and technologies needed to effect virtual integration
- Built on the three-legged stool of
  - Model-based,
  - Proof-Based, and
  - Component-Based engineering
- Structured/transformable data interfaces
- A global collaboration

**SAVI is not**

- A software tool or a design tool
- A continuation of current system development practices
Modified Business Model

System Integrator defines a new product using internal repository of virtual “parts”
Specifications for virtual subsystems sent to suppliers
Proposed and developed subsystem models incrementally provided to integrator
Virtual Systems Integration
Uncovers Errors Earlier in Development

Standarized architecture language with strong semantics, the Model Bus and Model Repository concepts in SAVI enable...

...early validation of system and embedded software system behavior to reduce integration errors.
## Architecture Design Language Requirements for Supporting Embedded Software System (ESS) Analyses

<table>
<thead>
<tr>
<th>Desired quality</th>
<th>Reason</th>
<th>AADL</th>
</tr>
</thead>
</table>
| ESS architecture concepts with precise semantics                               | Standardized analysis quantitative assessment                           | ESS abstractions as language primitives  
Semantics well-documented for each component & interaction category |
|                                                                                | Use of formal methods                                                  |                                                                                                                                 |
| Checkable consistency of architecture formation                                 | Incremental change impact detectable  
Impact analysis across quality attributes                                    | Compilable strongly typed language with standard legality & consistency rules  
EMF-based meta model drives XMI standard  
Design & operational quality attributes |
| Component-based fidelity multi-dimensional modeling                             | Consistency & quantitative analysis early & throughout development life cycle | Hierarchical composable SW/HW/physical components with interaction behavior & timing  
Explicit support for templates, patterns, incomplete models  
Standard extensions via property sets & annex sublanguages to core |
| Model scalability, variability & management                                     | Large scale system modeling & subcontractor management                  | Spec/instance separation  
Type/implementation variation  
Dynamic re-configurability  
Packages to manage model space |
## PoC Prioritized Requirements

<table>
<thead>
<tr>
<th>#</th>
<th>Requirement</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Establish Model Bus infrastructure</td>
<td>Process</td>
</tr>
<tr>
<td>2</td>
<td>Establish Model Repository Infrastructure</td>
<td>Process</td>
</tr>
<tr>
<td>3</td>
<td>Inform RoI estimates through POC performance &amp; results</td>
<td>Process</td>
</tr>
<tr>
<td>4</td>
<td>Analyses be conducted across the system</td>
<td>Analysis</td>
</tr>
<tr>
<td>5</td>
<td>Two or more analyses must be conducted</td>
<td>Analysis</td>
</tr>
<tr>
<td>6</td>
<td>Analyses be conducted at multiple levels of abstraction</td>
<td>Analysis</td>
</tr>
<tr>
<td>7</td>
<td>Analyses must validate system model consistency at multiple levels of abstraction</td>
<td>Analysis</td>
</tr>
<tr>
<td>8</td>
<td>Analyses must be conducted at the highest system level abstraction</td>
<td>Analysis</td>
</tr>
<tr>
<td>9</td>
<td>Model infrastructure must contain multiple model representations</td>
<td>Model</td>
</tr>
<tr>
<td>10</td>
<td>Model infrastructure must contain multiple communicating components</td>
<td>Model</td>
</tr>
</tbody>
</table>
AVSI SAVI Work Packages
As-is and To-be development process
Error & ROI Model
Model repository & bus infrastructure
Commercial tool integration
Standards based process
Acquisition Certification

Benefits of System Architecture Virtual Integration (SAVI)
Sensitivity analysis for uncertainty
Confidence in implementation

Predictive
Validated

Requirements Engineering
System Design
Software Architectural Design
Component Software Design

High-level AADL Model
Detailed AADL Model
Specify Model-Code Interfaces

Code Development
Integration Test
System Test
Acceptance Test

→ generation of test cases
← updating models with actual data

Model-driven artifact generation
Conformance of models and systems
SAVI Proof Of Concept Demo

Incremental Multi-Fidelity Multi-dimensional Multi-Layered Architecture Modeling & Analysis

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

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

Additional Opportunities:
- Safety & security analysis
- Fault modeling & impact analysis
- What-if trade studies

System & software system
Integrator & subcontractor virtual integration
Proof-of-Concept Demonstration - (4/4)

- Did the results from this PoC Demonstration indicate that the System Architecture Virtual Integration (SAVI) methodology is technically feasible to pursue?

**UNANIMOUS -- YES!**

- Core concepts were demonstrated on three different models, **BUT**...
  - Scalability was not fully explored
  - Open issues with AADL (ADL used in PoC) are to be explored
    - *Meets needs of all Use Cases?*
    - *Full compatibility with DoD AF Version 2?*

June 09
Cost Reduction through Rework Avoidance

Cost reduction ranges from $717M (7.8%) to $2,391M (26.1%) on a $9,176M new airplane project (2014-2018)

Every increase of 1% in defect removal efficiency results in a conservative cost reduction of $22M

Estimates based on conservative assumptions

- Based on industry data from SAVI participants
- Model assumes development of a single large aircraft in the 2014-2018 timeframe
- Savings largely driven by reduction of rework via discovery of requirements related problems earlier in the development lifecycle
- ROI does not include savings in maintenance & field upgrades, schedule overrun, loss of life & equipment, mission delay
- Conservative – used research investment of $108M, 2.5x expected, 2010-2014.
Spiral Development Planned

- Three Iterations to Reach TRL 9

- Schedule Roadmap Next
System components are captured as component abstractions & properties supported by detailed component models.
A Fault Propagation Use Case
System & Embedded Software Loop

The System

Hydraulic System
Pump failure due to SW

Avionics System
Unresponsive flaps affect flight control

The Software

Hydraulic Control Software
Pump control

Avionics Software
Auto-pilot
Flight control

Use of AADL Error Model Annex for Fault Propagation Modeling

AADL Model of Physical System, Application Software, Computer System

Computer System

Bus failure

System and Software Architecture Using AADL
Feiler, Nov 2010
© 2010 Carnegie Mellon University
4. To demonstrate the integration of multi-model representations within the EPoCD Architecture Framework AADL model.
AADL and Safety-Criticality

Fault management

- Architecture patterns in AADL
  - Redundancy, health monitoring, …
- Fault tolerant configurations & modes

Dependability

- Error Model Annex to AADL
- Specification of fault occurrence and fault propagation information
- Use for hazard and fault effect modeling
- Reliability & fault tree analysis

Behavior validation

- Behavior Annex to AADL
- Model checking
- Source code validation

Consistency checking of safety-criticality levels
AADL Error Annex

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.
Reliability Validation & Improvement Framework

From System Requirements to Software Requirements
Formalized requirements
Focus on safety-criticality requirements

Mission Requirements
Function
Behavior
Performance

Safety-criticality Requirements
Reliability
Safety
Security

System & Software Assurance
Sufficient justified confidence that mission & safety-criticality requirements (claims) are met
Evidence through reviews, analysis, testing, and validated assumptions

End-to-end System Validation and Verification

End-to-end System Validation and Verification

Model Repository

Architecture Model
Component Models
System Implementation

Resource & Performance Analysis
Reliability & Safety Analysis
Mode & Interaction Behavior Analysis

Static Analysis
Formal methods to complement testing
End-to-end V&V of mission and safety-criticality requirements

Architecture-centric Model-based Engineering
Architecture model with well-defined semantics (AADL)
Incremental validation through virtual integration
Consistency across analysis dimension

© 2010 Carnegie Mellon University
Incremental Architecture-centric Validation & Verification Improves Qualification Confidence

Build the System

Build the Assurance Case
AADL: Security Modeling

Confidentiality concerns that sensitive data should only be disclosed to or accessed/modified by authorized users, i.e., enforcing prevention of unauthorized disclosure of information.

Objective: Model security attributes for an architecture to verify that data is properly accessed and handled by users and applications.

Confidentiality frameworks

- Bell-LaPadula framework: military applications
- Chinese wall framework: commercial applications
- Access role/role-based access framework
- MILS
Model Based Analysis for Information Assurance

EDICT IA Vision

An integrated tool suite for the specification, design, evaluation and deployment of high confidence systems

- An innovative approach for
  - IA domain specific modeling and systematic evaluation and analysis
  - Integration with standard development and certification processes

- Utilize a Model Driven development approach to support the specification and evaluation of system properties throughout the system lifecycle

- Support for modeling and analysis of MILS design approaches

- Provide views and tools that are tuned to the needs of system stakeholders cross cutting concerns and activities
  - Architects – Security Engineers - Certifiers

- Utilize analysis after system deployment to support
  - Upgrades – Changes In Threat – Changes In Operations

Open Modeling and Tool Platform

- Eclipse Platform for tool portability and open integration
- AADL for system architecture modeling
- XML based information storage
Architecture-Centric Virtual Integration Impact

• Reduce the risks
  – Analyze system early and throughout life cycle
  – Understand system wide impact
  – Validate assumptions across system

• Increase the confidence
  – Validate models to complement integration testing
  – Validate model assumptions in operational system
  – Evolve system models in increasing fidelity

• Reduce the cost
  – Fewer system integration problems (SAVI ROI)
  – Fewer validation steps through use of validated generators
The SAVI demo video can be watched over the web at

www.aadl.info/aadl/savi/2009POCDemo/avsisaviPOCDemo35min.html

Design, Verification and Implementation of MILS Systems

Julien DELANGE, Laurent PAUTET

TELECOM ParisTech -- delange@enst.fr, pautet@enst.fr

Fabrice KORDON

LIP6, Univ. P & M. Curie -- fabrice.kordon@lip6r