Open Source AADL Workbench for Virtual System Integration
Peter Feiler
Oct 2015

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA  15213
Outline
Mission and Safety-Critical System Challenges

Virtual System Integration with SAE AADL

Samples of AADL Workbench Capabilities
We Rely on Software for Safe Aircraft Operation

Embedded software systems introduce a new class of problems not addressed by traditional system safety analysis.
### Safety-Critical System Challenges

#### Total System Cost
- **Boeing 777**: $12B
- **F-35**: $59B

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

#### 70% of faults introduce in requirements and architecture design
- 80% of faults discovered post unit test

#### Where Faults are Introduced

<table>
<thead>
<tr>
<th>Requirements Architecture Design</th>
<th>Code</th>
<th>Unit Test</th>
<th>Integration Test</th>
<th>Acceptance Test</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>70%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Where Faults are Found

<table>
<thead>
<tr>
<th>Nominal Cost Per Fault for Fault Removal</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.5%</td>
</tr>
<tr>
<td>16%</td>
</tr>
<tr>
<td>50.5%</td>
</tr>
<tr>
<td>9%</td>
</tr>
<tr>
<td>20.5%</td>
</tr>
</tbody>
</table>

#### 80% of faults discovered post unit test

**Sources:** Critical Code; NIST, NASA, INCOSE, and Aircraft Industry Studies.
Outline

Mission and Safety-Critical System Challenges

Virtual System Integration with SAE AADL

Samples of AADL Workbench Capabilities
AADL focuses on interaction between the three elements of a software-reliant mission and safety-critical systems.
Analysis of Virtually Integrated Software Systems

Single Annotated Architecture Model Addresses Impact Across Operational Quality Attributes

Safety & Reliability
- MTBF
- FMEA
- Hazard analysis

Potential new hazard

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

Affects temporal correctness

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

Increased latency

Architecture Model

Security
- Intrusion
- Integrity
- Confidentiality

Resource Consumption
- Bandwidth
- CPU time
- Power consumption

Higher CPU demand

Data Quality

Change of Encryption from 128 bit to 256 bit

Increased latency

Resource Consumption
Early Discovery through Virtual System Integration

Reduced cost through Early Discovery

80% Post Unit Test Discovery

Assure the System

Build the System
Outline
Mission and Safety-Critical System Challenges

Virtual System Integration with SAE AADL

Samples of AADL Workbench Capabilities
<table>
<thead>
<tr>
<th>Modeling Capabilities</th>
<th>Analysis Capabilities</th>
<th>Usability Capabilities</th>
<th>External Contributions</th>
</tr>
</thead>
<tbody>
<tr>
<td>AADL</td>
<td>Resource Budget</td>
<td>Context sensitive help</td>
<td>Resolute</td>
</tr>
<tr>
<td>EMV2</td>
<td>Latency</td>
<td>Role specific workflow</td>
<td>Agree</td>
</tr>
<tr>
<td>BA</td>
<td>Safety</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Type Consistency</strong></td>
<td>RMA/EDF Scheduling</td>
<td><strong>Expanded Navigation Views</strong></td>
<td>Ocarina Code Generation DeOS, VxWorks</td>
</tr>
<tr>
<td><strong>Semantic Consistency</strong></td>
<td>Resource Allocation</td>
<td></td>
<td>MAST Scheduling</td>
</tr>
<tr>
<td>Team Mgmt</td>
<td>Functional Integration</td>
<td><strong>Graphical Editor</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ARINC653 Support</td>
<td></td>
<td><strong>Independent contributions</strong></td>
</tr>
</tbody>
</table>

* aka Open Source AADL Tool Environment (OSATE)
Graphical Editing and Deployment View

Table-based property view and editing of selected component

Nested components
Partition binding
Context-Sensitive Editing

Type-sensitive Data Entry

Content Assist & QuickFix
End to End Latency Analysis

Latency analysis throughout life cycle
• Functional & system architecture: latency budgets
• Task & communication architecture: processing, sampling, transfer
• Platform architecture: partitions, protocols, computer hardware

Latency contributors
• Systems: processing, sampling, queuing latency
• Connections: protocol overhead, physical transfer, sampling
• Partitions: sampling, window schedule

Trade studies
• Best-case & worst-case, latency jitter
• Mid-frame and frame-delayed communication
• Synchronous and asynchronous systems
• Partition end and major frame output policy
• Empty & full queue

Top-down & bottom-up
• Latency budgets & rate, size, time based actuals

Utilizes end-to-end flows
Incremental refinement
Interprets deployment bindings
Operational mode specific analysis
Latency Analysis Views and Results

Latency Analysis Table:

<table>
<thead>
<tr>
<th>Contributor</th>
<th>Min Specfic</th>
<th>Min Value</th>
<th>Min Method</th>
<th>Max Specfic</th>
<th>Max Value</th>
<th>Max Method</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition: cpu,part1</td>
<td>0.0ms</td>
<td>partition offset</td>
<td>0.0ms</td>
<td>partition offset</td>
<td>Initial 200.0ms partition latency not added</td>
<td></td>
<td></td>
</tr>
<tr>
<td>thread s1.ts</td>
<td>0.0ms</td>
<td>first sampling</td>
<td>0.0ms</td>
<td>first sampling</td>
<td>Initial 20.0ms sampling latency not added</td>
<td></td>
<td></td>
</tr>
<tr>
<td>thread s1.ts</td>
<td>1.0ms</td>
<td>processing time</td>
<td>2.0ms</td>
<td>processing time</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Partition: cpu,part1</td>
<td>199.0ms</td>
<td>partition output (MF)</td>
<td>198.0ms</td>
<td>partition output (MF)</td>
<td>Output at 200.0ms major frame</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connection</td>
<td>0.0ms</td>
<td>no latency</td>
<td>0.0ms</td>
<td>no latency</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Partition: cpu,part3</td>
<td>100.0ms</td>
<td>partition offset</td>
<td>100.0ms</td>
<td>partition offset</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>thread p.tf</td>
<td>0.0ms</td>
<td>sampling</td>
<td>0.0ms</td>
<td>sampling</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>thread p.tf</td>
<td>0.2ms</td>
<td>processing time</td>
<td>0.3ms</td>
<td>processing time</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Partition: cpu,part3</td>
<td>98.0ms</td>
<td>partition output (MF)</td>
<td>97.0ms</td>
<td>partition output (MF)</td>
<td>Output at 200.0ms major frame</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connection</td>
<td>0.0ms</td>
<td>no latency</td>
<td>0.0ms</td>
<td>no latency</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Partition: cpu,part4</td>
<td>150.0ms</td>
<td>partition offset</td>
<td>150.0ms</td>
<td>partition offset</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>thread a.tc</td>
<td>0.0ms</td>
<td>sampling</td>
<td>0.0ms</td>
<td>sampling</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>thread a.tc</td>
<td>1.0ms</td>
<td>processing time</td>
<td>3.0ms</td>
<td>processing time</td>
<td>Task period smaller than partition period</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Immediate Connection</td>
<td>0.0ms</td>
<td>no latency</td>
<td>0.0ms</td>
<td>no latency</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>thread a.td</td>
<td>0.0ms</td>
<td>processing time</td>
<td>0.0ms</td>
<td>no latency</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>thread a.td</td>
<td>1.0ms</td>
<td>processing time</td>
<td>2.0ms</td>
<td>processing time</td>
<td>Task period smaller than partition period</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Latency Total</td>
<td>0.0ms</td>
<td>552.0ms</td>
<td>0.0ms</td>
<td>555.0ms</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>End to End Latency</td>
<td>20.0ms</td>
<td>30.0ms</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

End to end Latency Summary:

- **ERROR**: Minimum actual latency total 552.0 ms exceeds expected maximum end to end latency 30.0 ms
- **ERROR**: Maximum actual latency 555.0ms exceeds expected end to end latency 30.0 ms
Advanced Scheduling Capabilities

Multicore Schedulers

• Rate-Monotonic with Memory Partitioning
• Global Earliest-Deadline-First (GEDF) Scheduler for Parallelized Tasks
• Memory Profiler for Multicore Processors
• GEDF for Parallelized Task with Memory Partitioning

Mixed-Criticality Scheduling (Zero-Slack Rate Monotonic)

• Asymmetric protection: protect high-criticality tasks from lower-criticality but allow higher-criticality to steal CPU cycles from lower-criticality
Rate Monotonic with Memory Partitioning

![Diagram of a computer system with memory hierarchy and partitioning]

**Classical iterative response-time test**

\[
R_{i+1}^k = C_i + \sum_{\tau_j \in h_p(\tau_i)} \left\{ \frac{R_j^k}{T_j} \right\} \cdot C_j 
\]

\[
+ \min \left\{ H_i \cdot RD_p + \sum_{\tau_j \in h_p(\tau_i)} \left[ \frac{R_j^k}{T_j} \right] \cdot H_j \cdot RD_p, \quad JD_p(R_i^k) \right\}
\]

- **Request-Driven (RD) Approach**
- **Job-Driven (JD) Approach**

**Average over-estimates are 8%**

(13% for a shared bank)
Support of SAE ARP4761 System Safety Assessment Practice

- Functional Hazard Assessment (FHA)
- Failure Mode & Effects Analysis (FMEA)
- Common Cause Analysis (CCA)
- Probabilistic Reliability & Availability Analysis
- Fault Tree Analysis (FTA)
- AADL & Error Model V2
Architectural Security Verification

AADL Model of QuadCopter Software System

```
only_receive_decrypt(x : component) <=
   ** "The component " x " only receives messages that pass Decrypt" **
   forall (c : connection).
      (parent(destination(c)) = x) =>
         is_sensor_data(c) or only_receive_decrypt_connection(c)

only_receive_decrypt_connection(c : connection) <=
   ** "The connection " c " only carries messages that pass Decrypt" **
   let src : component = parent(source(c));
   unalterable_connection(c) and (is_decrypt(src) or only_receive_decrypt(src))
```

DARPA High-Assurance Cyber Military Systems (HACMS)
Secure Mathematically-Assured Composition of Control Models (SMACCM) Project
Towards an Architecture-Centric Virtual Integration Practice (ACVIP)

- Army and other Government Shadow Projects
  - Common Avionics Architecture System
  - Apache Block III ATAM
  - JPL Mission Data System
  - CH47F Health Monitor

- Future Vertical Lift
  - Architecture-centric Acquisition
  - JMR TD: ACVIP Shadow Projects
  - Virtual System Integration
  - System Assurance

- System Architecture Virtual Integration (SAVI) Software & Systems Engineering
  - AADL
  - Software & System Co-engineering
  - Multi-team Safety
  - Requirements Assurance

- SAE AADL Standard & AADL Workbench: Research Transition Platform
  - DARPA MetaH
  - ACME
  - AADL Error Model
  - European Commission SLIM/FIACT
  - DARPA META
  - DARPA HACMS Security

- US & European Research Initiatives
  - OMG MARTE Embedded Systems
  - ARINC653 Partitions
  - Avionics Network Standards
  - System Safety Practice Standards
  - Regulatory Guidance NRC, FDA, UL

- Other Standards and Regulatory Guidance

- 2004 to 2016
Contact Information

Peter Feiler/Lutz Wrage
Software Solutions Division
Email: phf/lwrage@sei.cmu.edu

Web
www.aadl.info
www.aadl.info/wiki

U.S. Mail
Software Engineering Institute
Customer Relations
4500 Fifth Avenue
Pittsburgh, PA 15213-2612
USA

Customer Relations
Email: info@sei.cmu.edu
SEI Phone: +1 412-268-5800
SEI Fax: +1 412-268-6257