ARINC653 AADL Annex

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA  15213

Julien Delange
07/08/2013
Context, Rationale

ARINC653
- Avionics standard
- Standardized API (called APEX – APplication EXecutive)
- Central part of the IMA philosophy
- Time & space partitioning

Rationale of ARINC653 annex for AADLv2
- Standardize modeling patterns
- Better modeling & analysis support
- Design associated toolset and framework
ARINC653 standard overview

Partitioning support
  - Software isolated in partitions
  - Partitions run as if they were on a single processor

Time isolation
  - Execution during a fixed & predefined time slice
  - Tasks scheduled with a dedicated scheduling policy

Space isolation
  - Code & data stored in a separated address space

Fault containment
  - Faults are propagated from processor to partitions
  - Partition-dependent recovery strategy
ARINC653 services

Time and space isolation
- Time slices allocation
- Address spaces allocation

Tasking (process) services
- Similar to the thread concept

Communication services
- Intra-partition
- Inter-partitions (module enforced)

Health Monitoring
- Recover faults at module, partition or process levels
Mapping ARINC653 to AADL

Partitioning support
- Partition execution context: virtual processor
- Partition content: process

Partitions control (with time & space specification)
- Support for partitions execution: processor

Tasking/process service
- Thread component

Communication services
- Rely on ports connections

Health Monitoring
- Dedicated properties (ARINC653 property set)
## Mapping rules

<table>
<thead>
<tr>
<th>ARINC 653 concept</th>
<th>AADL entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition</td>
<td>Process bound to virtual processor and memory</td>
</tr>
<tr>
<td>Module</td>
<td>Processor with virtual processor sub-components</td>
</tr>
<tr>
<td>Process</td>
<td>Thread</td>
</tr>
<tr>
<td>Intra-partition comm.</td>
<td>Data port (blackboard)</td>
</tr>
<tr>
<td></td>
<td>Event data port (buffer)</td>
</tr>
<tr>
<td>Inter-partition comm.</td>
<td>Data port (sampling)</td>
</tr>
<tr>
<td></td>
<td>Event data port (queueing)</td>
</tr>
<tr>
<td>Health Monitoring</td>
<td>Properties on components</td>
</tr>
</tbody>
</table>
ARINC653 system mapping example

System view

AADL view

Module

Partition 1

Partition 2

Process 1

Process 2

P1

P2

Partition 1

Partition 2
Tool support

Modeling support
Support for dedicated properties

Graphical view
Integration in the Instance Model Viewer

Analysis tool
Analyze and detect system issues from models

Third-party tools integration
Code generation for ARINC653 compliant systems
Modeling support

Integration of properties

Graphical view in IMV

```
processor implementation arinc_module.two_partitions
subcomponents
  runtime1 : virtual processor partition.i;
  runtime2 : virtual processor partition.i;
properties
  ARINC653::Partition_Slots => (10 ms, 20 ms);
  ARINC653::Slots_Allocation => (reference (runtime1),
    reference (runtime2));
  ARINC653::Module_Major_Frame => 30 ms;
end arinc_module.two_partitions;
```
Model analysis

System Performance
- Latency
- Architecture refinement

Architecture Consistency
- Scheduling definition
- Partitions isolation
- Resources dimension
- Impact between partitions having heterogeneous criticality levels

Easily extensible
- Implementation with LUTE, a constraint language from Rockwell-Collins
Model analysis - Latency

Timing concerns
- Thread behavior
- Runtime behavior
- OS specific properties

Migration impact
- Federated vs. IMA
Model analysis – Criticality (safety/security)

Crit. sender $\equiv$ Crit. receiver

### Theorem Safety

```
foreach Conn in Connection_Set do
    foreach P_Src in {x in Process_Set | Owner(Source(Conn)) = x} do
        foreach P_Dst in {y in Process_Set | Owner(Destination(Conn)) = y} do
            foreach Runtime_Src in {w in Virtual_Processor_Set | Is_Bound_To(P_Src, w)} do
                foreach Runtime_Dst in {z in Virtual_Processor_Set | Is_Bound_To(P_Dst, z)} do
                    check
                    ((Property(Runtime_Src, "ARINC653::Criticality")) =
                    (Property(Runtime_Dst, "ARINC653::Criticality")));
                end;
        end;
    end;
end;
```
Model analysis – Memory dimensioning

Partition requirements

- Thread Stack
- Code and data
- Context/env data

\[ \leq \]

Associated segment size

```
thm check_memory_requirements_partitions
  foreach prs in Process_Set do
    Thrs := \{x in Thread_Set | Is_Direct_Subcomponent_Of (x, prs)\};
    mems := \{x in Memory_Set | Is_Bound_To (prs, x)\};
    check
      \(((\text{Sum (Property (Thrs, "Source(Stack_Size")}) +
        \text{Sum (Property (Thrs, "Source(Data_Size")}) +
        \text{Sum (Property (Thrs, "Source(Code_Size")}))\}
      < \text{Sum (Property (mems, "byte_count")})\));
  end;
```
Third Party tools, code generation

Automatic ARINC653 configuration production (Ocarina)
  Generate ARINC653 XML configuration
  Reflect partitions and modules AADL requirements

Implementation Generation (Ocarina, RAMSES)
  Create ARINC653 C code
  Produce code from model, ensuring requirements enforcement
  With link to functional model (e.g. Simulink, SCADE, etc.)
Conclusion

Standardized modeling patterns for ARINC653 systems
  Support in OSATE
  Third-party support for implementation production

Analysis for several architecture candidates
  Capture IMA vs. federated architectures
  Problems when migrating from one execution platform to another

Extensible framework
  Implementation with LUTE
  Ease for extension and add new theorems
Links & Useful Information

AADL website – http://www.aadl.info

AADL wiki - https://wiki.sei.cmu.edu/aadl/index.php/Main_Page

ARINC653 AADL annex standard - http://standards.sae.org/as5506/2/

RAMSES - http://penelope.enst.fr/aadl/wiki/RAMSESInstallationSources
Contact

Presenter / Point of Contact
Dr. Julien Delange
RTSS AP Initiative
Telephone: +1 412-268-9652
Email: jdelange@sei.cmu.edu

Web
www.aadl.info
www.sei.cmu.edu
www.sei.cmu.edu/contact.cfm

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

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